£3.20

.EXE

OCTOBER 1990 VOL 5 ISSUE 5

The Software Developers' Magazine

Security comes to the fore. Codes and secrets galore this month.

Dongles and Software protection: Which one to choose - if any.

Don't take this mag through US customs! We explain the DES encryption algorithm.

Darrel Ince explains how to use a PC to win a Nobel prize.

The Third Side reaches POP-11: like a LISP without tears.

THE TRAINING LEADERS

Applications and Systems Programmers

What comes after C?

C++ is fast emerging as the next step forward for today's C developer.

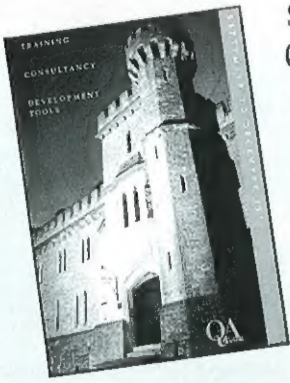
QA's C++ courses are specially designed for you to make the most effective transition to C++ in the minimum time. They combine excellent course materials and tutorials with practical hands-on experience. Covering not just the syntax of the language, but also the concepts of object oriented design.

And this is just part of QA's comprehensive C training.

A wide range of courses for the beginner, right through to the highly experienced systems programmer, ensure total training to the exacting standards required for project success.

In fact, there's no better way than QA to keep up-to-date with current and future C based developments.

So for further details make sure you post the coupon today. Or simply call Samantha Trinder on 0285-655888.



'C' COURSE OPTIONS

ALSO CONSULTANCY

Microsoft Authorised

glockenspiel APPROVED

Please complete and mail to QA Training, Cecily Hill Castle, Cirencester, Glos GL7 2EF.

Please send me details on QA's ☐ C courses ☐ C++ courses ☐ C consultancy.

Please send me details on QA's courses in: ☐ OS/2 and PM ☐ Windows ☐ Systems Support ☐ Networking ☐ DOS ☐ UNIX.

NAME | TITLE

COMPANY

ADDRESS

POSTCODE

TEL. NO.

CIRCLE NO. 241

QA TRAINING

Vol 5 Issue 5

.EXE

October 1990

Editor: Willie Watts
Editorial Assistant: Wendie Leek
Display Advertising Manager: Sandra Inniss-Palmer
Display Sales Executive: Ed Butcher
Recruitment Sales Executive: Jeremy Cox
Design & Layout: Katerina Adams, Mark English
Reader Services & Promotions: Helena Adams
Office Administrator: Carla Pearce
Publisher: Jon Howell
Front Cover Illustration: Ian Wright

General

.EXE Magazine is independent and not affiliated to any vendor of hardware, software or services. It is published by Process Communications Ltd, 10 Barley Mow Passage, Chiswick, London W4 4PH.
Tel: (Advertising/Editorial/Production) 081 994 6477
(Subscriptions) 044282 4501
Facsimile: 081 994 1533 Telex: 8811418 SPACES G
ISSN: 0268-6872

Subscriptions

.EXE Magazine is a monthly journal for software developers. It is available only by subscription, at a cost of £35 per annum (11 issues) in the UK (see insert card for academic and overseas rate). 'A subscription implies that this journal will be sent to the subscriber until one of the three expires' - AG Macdonell. The magazine is published around the 26th of the month preceding the cover date. There is no January issue. To subscribe, please call 044282 4501, or write to The Subscriptions Manager, .EXE Magazine, 10 Barley Mow Passage, Chiswick, London W4 4PH. We can invoice your company or take ACCESS and VISA cards. In case of a query about your subscription, please call 044282 4501.

Back issues are available at a cost of £3.50 each - call our Chiswick office (081 994 6477) for a list of issues still in print. We also offer a range of magazine binders, disk holders etc - please call for details.

Editorial

Editorial enquiries should be addressed to The Editor, .EXE Magazine, 10 Barley Mow Passage, Chiswick, London W4 4PH. We welcome letters, opinions, suggestions and articles from our readers. If you are interested in contributing articles, please write to this office for a copy of our Contributors' Guide.

Information contained in .EXE is believed to be correct. If errors are found, we will endeavour to publish a clarification in the next available issue.

From time to time, we offer to copy the PC program code described in an article onto diskette for our readers. In this case, please send a blank, formatted disk with a self-addressed, prepaid mailer to the editorial address given above. We can copy both 5.25" and 3.5" disks.

The publishers can accept no liability for any consequences of using software distributed in this way.

Advertising

If you are interested in advertising in this magazine, please write to the Display Advertising Manager/Recruitment Sales Manager (as appropriate), .EXE Magazine, 10 Barley Mow Passage, Chiswick, London W4 4PH, or call 081 994 6477 for details of our advertising rates.

Pronunciation

The name of .EXE Magazine is pronounced to rhyme with 'not sexy magazine'.

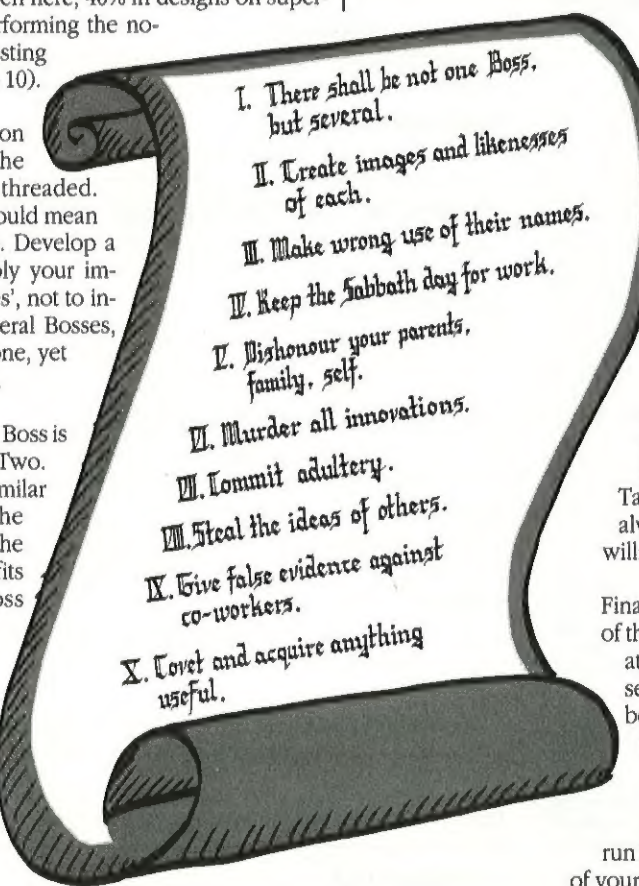
Copyright

Special Issue - Security

Unary survival - Ten Tenets

The economic recession threatens job security, both contract and permanent. William Campbell provides some axioms for surviving the cuts.

Once again a downturn in the computer industry is having an adverse impact upon employment. For some, destiny's flow chart will mean an unconditional `GOTO` or an `exit()`. But for those who employ a survival code utilising these 'Ten Tenets', the cutbacks need not result in a job `abend`. Programming efficiency or brilliance in design have zero effect in preserving your job. The ability to stay resident comes from expertise in the user interface; an awareness of the Company, and its people. Knowing when to pop up with the pat on the back, or with the knife in the back. Application of these Tenets will require time and effort. The 40-20-40 rule applies even here; 40% in designs on supervisors (Tenets 1 to 3), 20% on performing the notional work (Tenet 4) and 40% testing co-workers' fallibilities (Tenets 5 to 10).



Tenet One advocates not to rely on your single manager, even if the Company is structurally single threaded. Hooking your job onto one boss could mean the lifeline becomes your garrotte. Develop a Personal Network of Bosses. Apply your imagination to cultivating 'Boss Nodes', not to interpreting specifications. With several Bosses, you will not seem a threat to any one, yet each will be a server to your cause.

A simple method of ingratiating the Boss is by the use of imitation - Tenet Two. Drink the same drinks, employ similar mannerisms, mimic knotting the tie/applying blusher. Reinforce the self image of each Boss Node. Benefits of scale accrue above four 'Boss Nodes'. Each will be delighted to hear scandal about another for their own use, yet treat you as confidante to their own problems. The Third Tenet extols the virtues of denigrating the others while flattering the one who is present. The slander and gossip is information and connections to protect your survival.

Weekdays are for tuning your Personal Network, including evening work, off-site, in-pub. Sunday was invented for work, that list of complaints or justification for slippage. Tenet Four - do any necessary work on the Sabbath, unless enhancing the Personal Network with squash or Schönberg.

Development of Boss dependency on you is a priority. It must be equally balanced by encouraging sympathies towards you, at the expense of sympathies to your co-workers. Not all of us are blessed with having one lung or hereditary debts, so extrapolation of the facts is demanded. Dead parents are OK, but only as a short lived theme. Their internecine separation has much more potential as the Fifth Tenet - dishonour anybody to gain sympathy.

To avoid any cause for your dismissal, do nothing, and allow nothing to be done. Tenet Six - kill all initiative and innovation. Keep your activities suspended and you will not be time-sliced. Unfortunately code must be written or a project planned. Tenet Seven supports Six, urging the adulteration of all requests and specifications. Rehash and reject until the pure original is grossly debased, and thereby delayed or deferred. Today's favoured project is tomorrow's disaster, so avoid all direct association with it. But if there is some kudos, it must be yours rather than a co-worker's.

This brings us neatly to Tenet Eight. Steal the work of others. Change it sufficiently to give you options of ownership and of dissociation. If the project is successful, it is due to your creativity and skill; any failure is attributable to the unstructured mess or stifled resources with which you had to heroically battle.

The Ninth Tenet encourages the reporting of co-worker failings and foibles to receptive ears. Quirks and incapabilities should be assiduously recorded in your database. Spontaneous opportunities for slander are few, so your creativity must be employed to steer conversations onto your target. Take care not to appear malicious, and always ensure the listener has both the willingness and the means to act.

Finally, the last Tenet encapsulates the mien of the rest. By living these nine rules you will attune your mind to recognising the possessions of others which would be of more benefit to you. Grab them. They are worthless in the guileless hands of others. A desk in a key position, earmarking or ear-biting the manager's daughter, enough phones, faxes and comms to run NATO, shall be yours, if it defers the date of your dismissal or termination.

A Quick Reference Guide to the Ten Tenets is shown in the centre panel. Maybe there is an echo of a previous Top Ten. Tenet Eight exhorts such plagiarism, so no problem there. Unless the original author objects, and plans some form of retribution.

EXE

William Campbell has been employed in computing since 1968, primarily on mini, micro and network systems, the last 10 years as a freelance Consultant/Programmer.

We seek contributors for this column. If you have an idea for a suitable (attitudinised, readable, short) article, please contact the Editor at the address given on page 1.

DEBUGGING TOOLS

Multiscope	OS/2	£260
MagicCV	386&PC-DOS	£195
Periscope I/512K	PC-DOS	£580
Periscope II	PC-DOS	£115
Periscope II-X	PC-DOS	£ 95
Quaid Analyser	PC-DOS	£125
Soft-Ice	386&PC-DOS	£360
Soft-Ice + Magic CV	386&PC-DOS	£490
Turbo Debugger v2	PC-DOS	£105

ADA COMPILERS

The new AdaStudent is an affordable way to learn this important language.

AdaGraduate	PC-DOS	£625
AdaVantage Dev Kit(R14)	PC-DOS	£1500
AdaVantage Prof Dev Kit	PC-DOS	£2495
AdaStudent	PC-DOS	£ 75
Janus/Ada MS-DOS comp'lr	MS-DOS	£355
IntegrAda	PC-DOS	£645

FORTRAN COMPILERS

The latest Prospero Fortran now supports OS/2. A new version of Lahey's compiler with improved optimisation now here.

Lahey F77L v4.02	MS-DOS	£435
Lahey F77L-EM/32	ergo+PC-DOS	£670
Lahey Personal Fort-77	PC-DOS	£ 79
FTN77/386 Fortran	PC-DOS	£695
RM/FORTRAN 77 v2.43	MS-DOS	£450
MS-FORTRAN 77 v5.0	OS/2&MS-DOS	£250
Prospero Fort 2.10	OS/2&MS-DOS	£240
Pro Fortran for GEM	MS-DOS	£ 80
Prospero PC Fortran	PC-DOS	£ 80
Utah Fortran	MS-DOS	£ 35
FS-Fort.(CGA & Herc)	MS-DOS	£ 34
SVS Fortran 386	PC-DOS	£575
Watfor 77	PC-DOS	£290
Pro Fortran v2.1	CP/M-86	£240
Pro Fortran v1.25	CP/M-80	£240
Nevada Fortran v3.3	CP/M-80	£ 35
Prospero Fortran (GEM)	ATARI ST	£ 80

THE C LANGUAGE

Microsoft C V6 is a complete rewrite with improved optimisation and a new Programmer's Workbench
High C V1.6 has been considerably improved, with better Microsoft C compatibility, and new documentation

C COMPILERS

Avocet C	PC-DOS	£210
Aztec C86 Professional	MS-DOS	£120
Aztec C86 Developer	MS-DOS	£175
Aztec C86 Commercial	MS-DOS	£290
CI C86 Plus v1.3	MS-DOS	£290
ECO-C88 V4.0	MS-DOS	£ 70
HIGH C (Metaware) v1.6	MS-DOS	£405
HIGH C 386 v1.6	PL386&MS-DOS	£605
HIGH C 386 v2.3	PL386&MS-DOS	£675
Hi-Tech C	MS-DOS	£135
Lattice C v6	OS/2&MS-DOS	£175
Let's C v4	MS-DOS	£ 60
Microsoft C v6	OS/2&PC-DOS	£269
Microsoft QuickC v2.5	PC-DOS	£ 60
M'soft QuickC/QuickAsm	PC-DOS	£105
MIX C	MS-DOS	£ 20
MIX Power C & lib s'ce	PC-DOS	£ 29
Turbo C v2	PC-DOS	£ 70
Turbo C++	PC-DOS	£120
Topspeed C for OS/2	OS/2	£335
Topspeed C (std)	PC-DOS	£135
Topspeed C Extended	PC-DOS	£270
Watcom C v8 std	MS-DOS	£269
Watcom C v8 Prof	OS/2&DOS	£335
Watcom C/386 std	PL386&MS-DOS	£625
Watcom C/386 Prof	PL386&MS-DOS	£830
Zortech C v3	PC-DOS	£ 48
Zortech C++ v2.1	PC-DOS	£120
Z'tech C++ v2.1 Devlper	PC-DOS	£270
Aztec C86 Developer	CP/M-86	£245
Aztec C86 Personal	CP/M-86	£140
Hi-Tech C	CP/M-86	£150
Lattice C	CDOS	£380
HIGH C (Metaware)	Flex OS 286	£405
Avocet C	CP/M-80	£210
Aztec C Personal 1.06D	CP/M-80	£120
Aztec Commercial 1.06D	CP/M-80	£200
Hi-Tech C	Z80 + CP/M-80	£ 99
Mix C	Z80 + CP/M-80	£ 35
Toolworks C/80 v3.1	CP/M-80	£ 45
Aztec C65 V1.05	Apple DOS	£200
Aztec C68K MPW C	MACINTOSH	£110
Aztec C68K C	MACINTOSH	£ 75
Aztec C68K/AM Developer	AMIGA	£175
Aztec C68K/AM Prof'snl	AMIGA	£120
Aztec C68K/ST Developer	ATARI	£175
Aztec C68K/ST Prof'snl	ATARI	£120
Hi-Tech C	ATARI	£ 99
Lattice C v3.04	ATARI	£ 85
Mark Williams C v2	ATARI	£110
Laser C	ATARI	£135
Prospero C	ATARI	£105

C CROSS COMPILERS

We supply 2500AD, Avocet, Aztec, Lattice, IAR and Hi-Tech Cross Compilers hosted on MS-DOS and targeted on Z80, 6502, 6801, 68HC11, 6301, 6809, 7811, 8051, 8096, 68000 & 68020.
Please call for information or advice.

C INTERPRETERS

Instant C/16M supports large programs on 286/386 machines.

C-terp V3.0X	PC-DOS	£165
Instant C/16M	PC-DOS	£725
Interactive C	PC-DOS	£195
Introducing C	PC-DOS	£ 85
Living C Plus	PC-DOS	£135
RUN/C Professional	MS-DOS	£110

CROSS ASSEMBLERS

We supply cross-assemblers by Avocet, 2500AD,Crash Barrier (METAI),IAR Systems and Pecan hosted on MS-DOS, CP/M-86 and CP/M-80 with over 30 target processors. In total over 200 products with no space to list them here. We hold some stock but you should allow 10-14 days for delivery.
Please call for information or advice.

C LIBRARIES

DATABASE/FILEHANDLING

Btrieve for OS/2	OS/2	£360
Btrieve v5.1	MS-DOS	£360
XQL	OS/2,PC-DOS	£460
Xtrieve Plus	OS/2,PC-DOS	£360
C/Database T'lchest sce	PC-DOS	£ 39
CBTREE	(source any C)	£135
C-Index/Plus	(source any C)	£175
C-ISAM (L,MS)	MS-DOS	£220
C-tree	(source any C)	£230
Essential Btree (s'ce)	PC-DOS	£120
CQL s'ce	PC-DOS	£270
Faircom Toolbox Prof	PC-DOS	£665
Pro-C & Workbench v2	PC-DOS	£585
R-tree	MS-DOS	£170
db-VISTA DBMS Object	PC-DOS	£450
Lattice dBCIII (L,MS)	MS-DOS	£185
Lattice dBCIII+	OS/2&MS-DOS	£365
SftFcus Btree&Isam	(sce any C)	£ 80
Zortech Btree/ISAM	PC-DOS	£ 75

GRAPHICS

Enhanced Graphics Tlkt	PC-DOS	£170
Essential Graphics v3	PC-DOS	£235
GFX Graphics	PC-DOS	£ 75
GraphiC v5 (CI,L,DS,MS)	PC-DOS	£290
graphics MENU/META v3	PC-DOS	£135
GSS Kernel system	OS2,PC-DOS	£350
GSS Graph Dev T'kit	OS2,PC-DOS	£350
HALO v3 (MSC5,LAT,TC)	PC-DOS	£230
HGraph (MSC)	PC-DOS	£ 80
Ingraf (MSC)	PC-DOS	£175
MetaWINDOW v3.6A	PC-DOS	£145
MetaWINDOW/Plus	PC-DOS	£190
MetaWINDOW/Premium	PC-DOS	£315
TurboWINDOW/C (Turbo)	PC-DOS	£ 79
QuickWINDOW/C (QC)	PC-DOS	£ 79

SCREEN & WINDOWS

Vitamin C for OS/2	OS/2&DOS	£240
Power Screen (MS&TC)	PC-DOS	£105
Blaise View Mngr.(s'ce)	PC-DOS	£245
Lattice Curses	PC-DOS	£105
Entelekon Windows(s'ce)	PC-DOS	£105
Multi-windows (MS,L)	PC-DOS	£190
Panel Plus II (source)	PC-DOS	£270
Vitamin C (source)	PC-DOS	£165
Vermont Views (MSC,TC)	PC-DOS	£345
Greenleaf Datawindow	PC-DOS	£180
Greenleaf Makeform	PC-DOS	£ 60
C-Scape(+s'ce&Look&Feel)	PC-DOS	£345
Zortech Windows(ZC++ZC3)	PC-DOS	£ 39
Curses/PC w. s'ce	PC-DOS	£265

GENERAL FUNCTIONS

C+O Class Lib (MS,TC)	PC-DOS	£155
C+O Cl's Lb (MS)WINDOWS&	PC-DOS	£155
C/Utilities T'chest sce	PC-DOS	£ 39
C-Worthy v1.2 + forms	MS-DOS	£340
Code Runner	PC-DOS	£105
PDK£1	PC-DOS	£ 75
Greenleaf Functions s'ce	PC-DOS	£135
G'leaf Super Functions	PC-DOS	£175
G'leaf Fin'cial Mathlib	PC-DOS	£230
The Heap Expander(MS&TC)	PC-DOS	£ 65
C Tools Plus/6.0 (MS)	PC-DOS	£105
Power Search	PC-DOS	£105
Turbo C Tools v2	PC-DOS	£105
Essential C Utils s'ce	PC-DOS	£145
Entelekon Funct(source)	PC-DOS	£105
Resident-C (MS,L)(s'ce)	PC-DOS	£145
Silverware C EMM Lib	PC-DOS	£115
WKS Library	PC-DOS	£160
Zortech Supertext	PC-DOS	£ 39

COMMS LIBRARIES

C Asynch Manager V3s'ce	PC-DOS	£135
Essential Comms(L,MS,T)	PC-DOS	£195
Greenleaf Comms(source)	PC-DOS	£205
Multi Comms (L,MS)	PC-DOS	£190
Silvercomm C Asynch	PC-DOS	£145
Zortech Comms (QC&TC)	PC-DOS	£ 65
Zortech Comms(ZC++&ZC3)	PC-DOS	£ 65

SCIENTIFIC LIBRARIES

C Lang Sci Lib v2	ANY C	£175
Mathpak 87 (L,MS)	MS-DOS	£ 85

PROGRAMMERS UTILITIES

PC-Lint	OS/2,MS-DOS	£ 85
C-Documentor	PC-DOS	£195
C-Scan	PC-DOS	£195
Clear+ for C	PC-DOS	£165
Lattice Comp Companion	MS-DOS	£ 65

REAL TIME & MULTI-TASKING TOOLS

Concurrent C (PC/MPX)sce	MS-DOS	£ 60
Multi C (L,MS,CI,TC)	PC-DOS	£190
Op.Sys.T'box (MS) s'ce	PC-DOS	£ 69
Timeslicer v3.01 (L)	PC-DOS	£165
Timeslicer v5 (MS5)	PC-DOS	£165
Over-C (L,MS)	PC-DOS	£225

PROGRAMMING TOOLS

Ada Compilers	Algol Compilers
Assemblers & Libs	AWK
Basic Compilers	Basic Interpreters
Basic Utilities	Basic Libraries
BCPL Compilers	C Compilers
C Interpreters	C Libraries
C Utilities	Cobol Compilers
Comms.Libraries	Cross Assemblers
Database Libs.	Debuggers
Dis-assemblers	Editors
Engineers Libs.	Expert Systems
Forth	Fortran Compilers
Fortran Libraries	Graphics Libraries
Icon	Linkers/Locaters
Lisp	Modula-2
Nial Interpreters	OPS 5
Pascal Compilers	Pascal Libraries
Prolog	Rexx
Risc	Screen Libraries
Smalltalk	Snobol

We stock many items for which there is no space in these advertisements.

PRICES & DELIVERY

Prices do not include VAT or other local taxes but do include delivery in UK and Europe. Please check prices at time of order, ads are prepared some weeks before publication. This page lists some of our products. Call us for a complete pricelist. Order by phone with your credit card.

DISK COPYING SERVICE

We can copy files to and from 600 disk formats including CP/M, CP/M-86, MS- DOS, PC-DOS, ISIS, APPLE, SIRIUS, BBC, TORCH, APRICOT, HP-150, TRSDOS, DEC RT- 11, IBM BEF, ATARI ST, AMSTRAD, MACINTOSH.
Our charge is £10.00 + disk + VAT with discounts on small quantities and disks are normally despatched within 24hrs of receipt. For more information call us.

GREY MATTER

2 Prigg Meadow, Ashburton, Devon TQ13 7DF

TEL. (0364) 53499

GREY MATTER

2 Prigg Meadow, Ashburton, Devon TQ13 7DF

TEL. (0364) 53499

GREY MATTER

2 Prigg Meadow, Ashburton, Devon TQ13 7DF

TEL. (0364) 53499

Modula-2 Conference
The first European Modula-2 Conference is to be held at the Polytechnic of Wales, Pontypridd, 17th-19th December. The guest speaker is to be Dr Roger Henry, convener of the ISO Modula-2 working group. To find out more, phone Dr M Al-Akaidi at the Polytechnic of Wales (0443 480480 x2271) or Steve Collins at Real Time Associates (081 656 7333).

Snow use
'Snow' is a WYSIWYG-style report writer which runs under lots of different operating systems and machines: MS-DOS, VMS, XENIX, UNIX etc. The latest release - V3.02c - supports indexed files produced by AcuCobol in addition to 100 odd others. Salford-based producer TSL is particularly proud of this feat, because of the wide portability of AcuCobol itself. The MS-DOS version of costs £395. More details from TSL on 061 745 7755.

Optimised IAR
IAR Systems has released a new version of its ANSI C cross-compiler. Targeted at the 8051 range of microcontrollers, the new compiler is equipped with an optimiser which, according to IAR, produces code that runs twice as fast as that produced by traditional compilers. The object code also supports various special hardware features, such as bank memory switching and A to D converters. The system is hosted on PCs. Contact IAR on 071 379 0344.

Take no RISC
It will develop 2 BIPS (Billion Instructions Per Second), contain 100 million transistors and four CPU units (including a specialised graphics section), and clock at a cool 250 MHz. Intel has been announcing its plans for future members of the 80x86 family - the specification given above is for the 80786, expected in the year 2000. The company also described the forthcoming 80586 (1992) and 80686 (1996) processors, by way of emphasising that it is not about to abandon CISC in favour of trendy RISC.

Could do better
IBM has introduced its PS/1 range of PCs to the UK. Based on the 80286, the machine consists of one box with 512 KB/1 MB of memory, a 3.5" floppy drive, an optional 30 MB hard disk and a VGA display. The machine is bundled with Microsoft Works, which IBM says is so easy for students to use that it is perfect for 'improving the quality of school reports.' I think *I know what it means...*

Glockenspiel endorsed

Glockenspiel has released V2.0 of the CommonView application framework for Microsoft Windows 3.0. The new software is supplied in the large memory model only - Glockenspiel claims that this will ease the difficulties of porting code to other GUIs. At the announcement, a Microsoft representative said that his company was aiming for source code compatibility between Glockenspiel's C++ and the MS product now under development - presumably an attempt to discourage MS C users, impatient for C++, wandering into the arms of Borland or Zortech. Glockenspiel is distributed in the UK by QA Training: 0285 655888.

Clipper V5.0 at last

To the enormous relief of pretty well everybody, Nantucket has finally shipped Clipper V5.0. As well as the new features noted in our August preview (preprocessor, more versatile variable declarations, pseudo object-oriented facilities for handling some user I/O and errors), Nantucket has incorporated a new virtual memory management system.

The VMM makes use of conventional memory, LIM V3.2+ expanded memory and hard disk swap space. It comprises of two distinct subsystems. The segmented virtual memory subsystem provides up to 64 MB of virtual address space. Database buffers are allocated within this system, up to the maximum available real RAM (conventional + expanded), which is 'usually 8 to 9 MB' *(sic)*. The separate object memory subsystem manages up to 16 MB of virtual memory, allocated within the segmented virtual memory subsystem (I hope you are following this carefully, I may set a test). Object memory contains character and array values generated by Clipper programs, and is garbage-collected. One apparent side-effect of this system is that, because it is not possible to switch off the VMM completely, all Clipper V5.0 applications run in non-LIM machines must be able to write to disk. The VMM is quite distinct from the bundled overlay system, based on Pocket Software's .RTLink linker, which allows large source code programs to be created.

Clipper V5.0 should be available now, priced £595 ex VAT. Registered users may upgrade for £150 + VAT and shipping; users who purchased after 12th June *1989* can upgrade for the price of shipping costs, which seems quite generous. Nantucket is on 0707 373600.

Strange bedfellows

ICE.TCP is a TCP/IP terminal emulation package for PCs; it does Wyse 60, VT220 and dear old ANSI. It comes with a Novell Netware shell drivers produced by Brigham Young University in the US. These drivers conform to Novell's software spec, but use Ethernet (= TCP/IP) format data packets to communicate. Load the BYU drivers onto every machine on the network, solder the bit of wire from the network to the bit of wire from your UNIX monster and, before you can say 'I don't think this is going to work', your workstations will be able to login to the Novell server, or UNIX machine, or both - without rebooting. Sounds neat, eh? Talk to ICE.TCP's UK distributor, Custom Business Systems Ltd (071 323 2297) for further details.

Paradox gets SQL

Borland has released V3.5 of its Paradox RDBMS. The upgrade has received the VROOMM treatment (Borland's proprietary memory manager), allegedly speeding up sorting and indexing 50% over V3.0. The system also includes something called Turbo Drive, which allows it to exploit up to 16 MB of extended memory, where available. You may now include PAL expressions in calculated fields of forms and reports. But the most interesting new feature is support for SQL.

To take advantage of SQL connectivity, you will need to purchase an extra module: the Paradox SQL Link. The current release can communicate with Microsoft SQL Server, Oracle Server V6.0 and IBM OS/2 EE Database Manager V1.2; in the pipeline are DEC's Rdb/VMS, DB2 and others. Paradox users can treat server databases as though they were native data, eg interrogating them with standard QBE statements. Programmers can access the server data by embedding SQL into PAL programs - PAL expressions may appear within the embedded SQL.

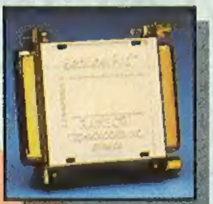
Paradox V3.5 plain vanilla costs £595, Paradox SQL Link is £325. If you have a network, then you will require Paradox Multi-Pack (replaces Paradox LAN Pack) which costs £895 and supports five users. I'm afraid I didn't understand the rules for how many copies of SQL Link per user you need on a network. Borland is hosting a series of corporate seminars in October in three cities across the country. The seminars will explain why client-server computing with Paradox is absolutely the thing, and will include speakers from IBM, 3COM and Oracle. To find out dates and places call 0734 333100.

S O F T W A R E   S E C U R I T Y

With £1 Billion Worth Of Protected Software...

Rainbow Is The Safe Bet.

SentinelPro™



• Runs under DOS, OS/2 and Xenix • Algorithm technique (Never a fixed response) • External parallel port installation • Minimal implementation effort • Higher level language interfaces included • 100 times faster than fixed-response devices (1 ms) • ASIC design for reliability

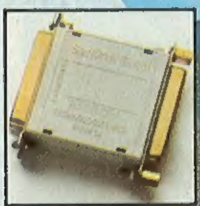
Sentinel-C™

• Protects multiple packages with one device • 126 bytes of non-volatile memory programmed before shipment of the software • Rainbow supplies a unique adapter for programming the unit • Higher level language interfaces included • Runs under DOS, OS/2 and Xenix • External parallel port installation

Eve™

• For the Macintosh SE and II • Complies with Apple Desktop Bus Interface requirements • Rainbow-assigned developer passwords to prevent tampering by other developers or sophisticated "hackers" • 7 locks per key, usable individually or in combination, on one or up to seven applications

SentinelShell™



• Runs under DOS on IBM PCs and compatibles • Protects without requiring access to the source code • Completely transparent to the end user • User-friendly software • Pocket-size key attaches quickly to any standard PC parallel port • ASIC design for reliability

DataSentry™

• Completely user-installable • Pocket-sized external device • Menu-driven, user-friendly interface • Single- or multi-user security system • Audit trail, log-on identifiers and automatic encryption/decryption of entire directories • Secures data transmitted by modems • Prevents recovery of data by utility programs

Software developers creating the latest applications for the IBM PC/XT/AT, PS/2 and compatible systems can now turn to the Software Sentinel range of hardware keys for the *first-class* in *world-class* software protection.

There's the best-selling SentinelPro, known worldwide for its virtually unbreakable security, its ASIC technology and its invisible operation.

A close relation, the Sentinel-C for custom configurations, enables multiple package protection with a single device.

For the Apple market, security-minded Mac software developers can now secure their return-on-investment, too. Eve plugs into the Mac ADB connector and is completely transparent to the user–providing up to seven programmable security locks per key.

Rainbow's latest protection strategy is the SentinelShell–that lets users place a "shell" around existing, off-the-shelf programs. Access can be limited to those issued a key, so that libraries, educational establishments and corporations can very simply guard their investments.

Available soon from Rainbow is the DataSentry, a user-installed key that provides low cost security for sensitive data in both database applications and corporate/banking environments.

No matter where you sell your software worldwide, stay in control of your distribution and revenue by choosing the internationally accepted standard in protection... Rainbow Technologies. Be sure. Protect your pot of gold at the end of the rainbow.

CALL TODAY OR ORDER NOW.

☐ Please send me a SentinelPro Evaluation Kit.
I enclose £50 + VAT Payable to Rainbow Technologies Ltd.
☐ Please debit my credit card. Access ___ Visa ___ Amex ___
Expiry Date ____________ (PLEASE TICK)

☐ Please send me more information.

Name ____________________

Position ____________________

Address ____________________

Telephone ____________________

Signature ____________________ .EXE 10/90

RAINBOW TECHNOLOGIES LTD., Shirley Lodge, 470 London Road, Slough, Berkshire SL3 8QY.

RAINBOW TECHNOLOGIES

Shirley Lodge, 470 London Road
Slough, Berkshire SL3 8QY
Tel: 0753-41512 Fax: 0753-43610

CIRCLE NO. 243

5 out of 5 hackers prefer other software protection methods to Hardlock E-Y-E®

What hackers dislike...
Hardlock E-Y-E was designed using cryptographic principles. It took the experience and know-how of Germany's No. 1 in software protection and the leading edge technology of a US semiconductor company to create the ultimate software protection tool. Hardlock E-Y-E is based on a custom chip featuring secure algorithmic response rather than simple bit swapping or counting schemes.

What software developers like...
Hardlock E-Y-E combines all the features software developers require in a single product: algorithmic response to provide security and an optional non-volatile memory to allow custom configurations. FAST Electronic has made implementation of Hardlock E-Y-E in your software easy. Use HL-Crypt to protect .EXE or .COM files, or incorporate high level language interface routines in your software. The algorithm parameters and the contents of the memory can be programmed in seconds using our Crypto-Programmer card. This unique card guarantees that no one else can burn your original codes. Simply plug the card into any PC slot and start up your own Hardlock E-Y-E workshop.

What your customers will like...
Hardlock E-Y-E allows unlimited backup copies. The device is shipped with the software for the user simply to plug into the parallel interface and forget. Daisy chainability, outstanding reliability (no battery is needed), and the most compact High-Tech design ensure that your customer will accept Hardlock E-Y-E.

What your accountant will like...
Hardlock E-Y-E needs no factory coding. This ensures optimum delivery schedules and stock flexibility. Revenues will go up as software piracy and multiple usage are prevented. Despite its wealth of features, Hardlock E-Y-E's prices remain competitive.

Hardlock E-Y-E
programmable, algorithmic response and memory option – all in one.

...As more and more software developers, customers and accountants appreciate the Hardlock E-Y-E device, hackers like it less and less.

MAGNIFEYE

Order your demo unit today. Contact Magnifeye,
235-239 Walmer Road, Walmer Studio #6, W11 4EY, Telephone 071 221 8024, Fax 071 792 3449.

CIRCLE NO. 244

Get out of Turbo Pascal

If all your accumulated software is written in Turbo Pascal, and you rather wish that it wasn't, then you may be interested in a new package from Maidenhead-based Migration Technology Ltd. The company, which is probably best known for a range of BASIC-to-C translators, has just brought out a Turbo Pascal to C translator.

The package can cope with Turbo Pascal V3.0 through to V5.0, and produces code suitable for compilation by TopSpeed, Turbo, Microsoft or generic ANSI C. The manufacturer claims that processed code runs faster because of fewer run-time checks, and of course C offers a much richer selection of third party libraries than TP; but the major advantage must surely be not having to rely on a proprietary compiler. The package retails at £395. Migration's number is 0628 668 511.

Mainframe APL

MicroAPL Ltd has produced a new version of its APL interpreter. APL.68000 Level II runs on a good selection of 68k based machines, from the Atari ST and the Apple Mac right the way down to multi-user UNIX boxes.

The new interpreter implements a version of the language based on IBM's APL2/370 standard. Enhancements over Level I APL include: mixed arrays, which can contain both character and numeric data; nested arrays; user-defined operators; multiple and selective specification; new primitive functions (I can't detail any of these, because they all involve APL's wonderful hieroglyphic symbols) system functions and commands.

Prices start at £170 for an Atari ST version, rising to £2500 for multi-user versions. Contact MicroAPL on 071 922 8866.

Heal thyself

PC-Technician is a hardware diagnostics package. It comes with a boot diskette, test diskettes, test plugs and even a carrying case, and was developed for use by field service engineers. Because it is loaded from bootstrap, it has various advantages over DOS-based systems. For example, it checks the memory into which it is about to be loaded, relocating if necessary. If the read/write head on the top side of your diskette drive is suspect, the program can be loaded using the head on the underside. The program deals with floppy and hard disk drives, video adapters, keyboards, parallel and serial interfaces, communications adapters and memory (base, extended and expanded). The pricing scheme varies according to processor and, as combination products are available, is as complicated as one of those cafe menus which list ham; ham and eggs; ham, eggs and chips; ham, eggs, peas and chips etc. Base price is £145.96 for an XT version, top is £930.24 for an XT/AT/386 combination product with 3.5" and 5.25" boot diskettes. The supplier is SJS Management Systems, on 0494 890257.

Injectable 1-2-3

PowerCell is a library for Microsoft C. It is also a complete implementation of a spreadsheet, broadly compatible with you-know-who, imitating standard keystrokes, menus, @ macro functions, file formats (WKS, WK1, DBF), macro language... in fact everything except graphics.

The idea is that you hook a little spreadsheet into your application. It is possible to control the amount of functionality (you can lose functions to reduce the application size), the size of the spreadsheet and the contents of the cells from C. Suppose you have a production control system. You write the part that schedules machines, personnel and materials for the production cycle. Via the spreadsheet, management can knock together all the reports that it likes using data that the C application pipes in. The user can be left happily mincing among his macros, freeing the programmer to get on with some work.

PowerCell is distributed by Systemstar (0992 500919). It costs £499 + VAT, or £1530 + VAT with complete source. You will require Microsoft C V5.1 or greater. There are no run-time royalties.

Turbo Analyst

Turbo Power Software, creator of the Object Professional Turbo Pascal Library, has released an updated version of Turbo Analyst, its toolkit for TP programmers. New features proffered include a revised Program Structure Analyser, which can produce reports showing the hierarchy of objects and detail the methods of each one. The PSA now uses extended memory, expanded memory and disk swap techniques, allowing it to analyse much larger programs. The Program Execution Timer and the Unit Info program have both been updated to cope with the new file formats of Turbo Pascal V5.5.

Turbo Analyst will doubtless filter through to the UK dealers in due course; but if you have trouble getting hold of it, then call Turbo Power direct on 0101 438 8608. US price is $99.

Reboot!

Reboot! is an MS-DOS utility from Ctrl Alt Deli . When run, it renames a chosen pair of files to AUTOEXEC.BAT and CONFIG.SYS, then reboots the computer. Just what you need if, for example, you use a big program like Ventura and a network but, owing to the size of the network driver, can't have both at once. This £29.95 utility will let you perform the switch/reboot painlessly - and, as it is menu-driven, it is suitable for end user operation.

Grammatik

Grammatik III was a brilliant, cynical program; an English style-checker which poured scorn on clichés, and all but spat with derision at fools who use words like 'paradigm'. The problem with it was that it only dealt in American English. Grammatik IV could cope with British English and even my word processor format (Sprint), but somewhere it had lost its anger, and let through all sorts of rubbish. I don't yet know if the new Grammatik Windows (£99 from Riva Ltd, 0420 22666) has regained the old spark, but it does cope with Word for Windows.

CXL change

We mentioned the CXL C function shareware library several months ago, but are still getting an occasional enquiry. All future questions, orders etc should be directed to the new owner, which is Innovative Data Concepts, 1657 The Fairways, Suite 101, Jenkintown, PA 19046, USA; telephone 0101 215 884 3373.

My Word!

Microsoft has just shipped Word for Windows V1.1, which is built to take advantage of Windows 3. If you run up Word for Windows V1.0 under Windows 3, some (but not all) of the controls have a curiously flat, distinctly 2D appearance. V1.1 presumably puts this right. But surely, if W for W had been written adhering to the Windows API (as Microsoft says we should), all the controls would have automatically been converted, without the need for a new version. Most mysterious.

Binary UNIX

AT&T's UNIX System Laboratories, Intel and SCO have agreed to define a common binary applications specification. Future versions of SCO and USL 386/486 UNIX will be able to run the same 'shrink-wrapped' software. Could this be the beginning of the end for C as the UNIX porting medium?

An Actor's life

The Whitewater Group's Actor product, an object oriented language and development environment for Microsoft Windows, has the reputation of being really good and easy to use for writing small programs, but running out of steam very quickly when you try and use it for larger applications.

UK distributors Neow Ltd has just introduced Actor V3.0, so perhaps this problem has now been tackled. Certainly the new version provides access to the full 16 MB of Windows memory, with a special memory manager system which allows applications sized up to 2 MB to run in a base 640 KB machine. There are also new classes to handle hierarchical menus and combo boxes - the latter are the new controls introduced with Windows 3.

ObjectGraphics is a brand new complementary product. According to Neow, it offers comprehensive support for the creation of 2D graphics, including various shapes, bitmaps and text. The appearance of images is adjusted with rendering tools such as 'pens' and 'brushes', which sounds rather fun, and the program also features control over drawing attributes such as line styles, fill patterns etc. ObjectGraphics is supplied with a sample application, ObjectDraw, which can itself be used to create graphics for Actor-based programs.

The other Whitewater upgrade to report is the WinTrieve ISAM library, which can be used with Actor or conventional programming languages. Actor costs £645, ObjectGraphics £325 and WinTrieve £325. Call Neow on 0628 668334.

Double your MB

A company called Perstor has sent me some information describing its range of PC hard disk controllers. These work with most standard hard disk drives, but store data using a proprietary coding method called Advanced Data Recording Technology. Using ADRT increases the number of sectors per track, and throughput of data from drive to controller, so in theory both disk capacity and performance are improved - Perstor claims 90% for the former and 10%-100% for the latter. The company emphasises that this is not a data compression technique.

I have details for two cards in the range. The 9008 HDC costs £189, supports MS-DOS V3.0+, MFM and RLL drives, requires an 8-bit slot and is suitable for 6 MHz to 12 MHz 8088-80286 systems. The PS180-16FN HDC costs £266, supports Novell, SCO Xenix, OS/2, MFM drives, requires a 16-bit slot and is suitable for 286 and 386 machines.

If you would like to find out more, call the UK distributors, Digitask Business Systems, on 029 377 6688.

BASIC Science

HTBasic is an MS-DOS version of Hewlett-Packard's HP ('Rocky Mountain') BASIC, as used on HP 9000 series machines. Favoured by scientists for laboratory work, it includes facilities for accessing IEEE-488 cards (IEEE-488 is a communications/bus standard frequently used by measurement devices and the like).

Version 3 of HTBasic offers improved integration with MS-DOS (an EXECUTE statement allows MS-DOS programs and commands to be run from within the environment), very close customisation of the environment to match Series 200/300 computer (so that many existing programs may run without alteration), extensive online Help and integral interrupt driven RS-232 communications. Existing features include device-independent graphics, complex mathematics, matrix arithmetic and recursion.

Prices for HTBasic start at £450 for the low end 286 version, up to £1000 for the 386/486 version (including an IEEE-488 card). Call Workstation Source, on 0628 75252, for details.

Greenleaf revises SuperFunctions

The SuperFunctions C library is a general-purpose MS-DOS package, containing functions to handle expanded memory, the mouse and text windows. You can also set up critical error handlers, and have numerous routines for time and date manipulation etc.

A new release contains additional functions for window manipulation: creating windows with 'shadows', scrolling text within windows and writing text using different colour attributes. There is also support for new compilers - Watcom C, TopSpeed C and Zortech C++ - plus use is made of Microsoft C V6.0's fast calling convention. The other compilers supported are Turbo C/C++, Lattice and QuickC. The US upgrade price is $45. Greenleaf is offering a free demo disk to interested parties; fax the company in the US on 0101 214 248 7830 (sorry, I can't give the phone number, as I only have a toll-free '0800' version, which is useless from the UK).

transIDRIS V4.0
Manx company Real Time Time Systems Ltd has released V4.0 of transIDRIS, its real time operating system for the transputer. Claimed enhancements include improved performance and response time, wider host support, new utilities and conformance to POSIX 1003.1 standards. The company is also particularly proud of its range of transputer cross-development tools, which allow the production of transputer object code without the target hardware. RTS is on 0624 661500.

DataEase extended
Version 4.2 of the DataEase RDBMS incorporates Rational Systems' DOS extender, allowing the creation of larger applications. There are also tweaks to the Multiform facility - better referential integrity features - and all round extended capacities. The price is £595; for details of the various upgrade deals call Sapphire on 081 554 4150.

4GL Race
The 4GL European Grand Prix is an annual competition at which a line-up of manufacturers compete against each other to build a set application. The competition, which is judged by a combination of time taken to produce the application and the quality of the result, was won by Sea-Change, running on a UNIX box. The unsuccessful competitors were Aspect (LANSA), Ingres, McDonnel (PRO-IV), Progress, System Builder (SB+), Systemator and Unify Corp (UNIFY).

GoScript
GoScript is an MS-DOS based PostScript interpreter which has just reached V3.0. It comes with 13 fonts (with 35 more available for an extra £90) and drivers for all the main types of non-PostScript printer (Epson FX and LQ, HP Laserjet etc). The £89 program also drives EGA/VGA graphics, so you can test your efforts on screen. Speed might prove a problem, though according to the distributor (David Pollard Associates - 0865 240048) with a 386 PC, print speeds are comparable with an Apple LaserWriter.

Windows User Group
An independent [Microsoft] Windows User Group has been formed. Stated intentions include providing hot-line telephone support, producing a regular colour magazine and compiling directories of Windows applications. More information from chairman Tim Bunning on 0909 501351.

Letters

We welcome short letters on any subject that is relevant to software development. Please write to The Editor, .EXE Magazine, 10 Barley Mow Passage, Chiswick, London W4 4PH. Unless your letter is marked 'Not for Publication', it will be considered for inclusion on this page.

Cowboy Builder

Sir,

I wish to draw the attention of your readers to a US product called 'The Builder' which allegedly compiles batch files. It is intended to be used for the creation of installation programs and the like.

The product did not work as per spec, or sufficiently near it to be useful, so I attempted to contact the manufacturers (from whom I had bought it direct). I was astonished at the level of technical support and quality of this product. I have both faxed and written to the publishers (Hyperkinetix of California) asking for information and have received not a peep in response. I ended up demanding: 'Am I stupid or should I ask for my money back?' The company did not deign to reply.

I subsequently returned the product and asked for my money back anyway. The company has continued to ignore me. Consequently, I find it hard to recommend this company.

Would you be interested in compiling a league table of the most totally useless and the most wonderful support organisations?

Ian Butterworth
Quin Butterworth Spangenthal Limited
London

C++ fuss

Sir,

I read the 'Soapbox' article in the August edition of .EXE with irritation. John Daniels implies that the only worthwhile reason to move from C to C++ is to make use of reusable code. As a compiler writer, I am only too aware of the problems associated with the design and maintenance of complex software, and my colleagues and I intend to use C++ for future projects. To my mind, the most valuable feature of C++ is the ability to write code without making irrevocable decisions about data structures and to do this without any loss of efficiency. The inline member functions and inline overloaded operators offer the programmer the opportunity to write in an elegant, expressive style without sacrificing anything in terms of efficiency.

I suspect that John Daniels is a supporter of the other object oriented languages. He cites the 'problem' with C++ that the compiler must know the memory size of each class at compile time. I think this is an irrevocable consequence of the decision that C++ should be as efficient (if not more so) than C. This decision is surely correct. Computer science is littered with samples of elegant languages which failed to make it because they could not be implemented efficiently.

Our industry is also obsessed with buzz words which blind some people to common sense. Thus John Daniels can write 'Object oriented design is still in its infancy. It cannot be comfortably integrated with other analysis of design approaches but requires, to maximise its benefits, that the whole software development life-cycle be revised' - This is obscurantist nonsense.

I have not found any reason to redesign my software development cycle - nor I suggest have most other enthusiastic users of the language.

Dr D Bailey
University of Salford
Manchester

Sir,

I am not a professional writer, however, if I were, then one of the guidelines I would adhere to (among others), would be to avoid using indecorous phraseology. You, your sub-editors or your 'guest writers' have manifestly failed in this respect.

I refer to the article titled 'OOP is more than using C++' which has the subheading 'As the OOP bandwagon moves onwards, developers are increasingly turning to C++ to take advantage of this *paridigm...*'. Thanks to WordPerfect V5.1 (I thought this letter would stand more chance of being printed if there was a technical reference), I discovered that the word should actually be **paradigm**. Unfortunately, WordPerfect does not define words.

Turning to the Concise English Dictionary (to look up the **correct** spelling this time) I find that **paradigm** is defined as '... an example of a word in it's grammatical inflections ...' Does this refer to the 'word' OOP or C++? (Assuming you believe that acronyms and words are synonymous.) Either way, I still don't quite understand what the sentence means.

For the sake of all the readers who subscribe to your magazine (as well as those, like me, who borrow it), could you please explain.

Finally, it is a criticism levelled at the computing press in general that they love using and inventing jargon. This is certainly true - but please don't compound the problem by using obscure and misspelled words.

RM Coleman
Cheltenham
Gloucester

The spelling mistake and use of crummy English are both down to me (WRW); please accept my apologies. I cannot resist trying to squeeze in words which seem to have too many consonants at the end - 'indict' is another good example.

However, I think that some of Mr Coleman's troubles are caused by the fact that he has a bum dictionary. Apart from the suspicious, incorrect use of it's, *it omits the most common (and intended) use of* paradigm, *which is 'a pattern or example' (verified in the Concise Oxford and Penguin English). I indict Mr Coleman for not completing his research before reaching down his Basildon Bond.*

May's last spark?

Sir,

I believe that Jules May will have to switch to C. He recently purchased a copy of the Microsoft Windows Development Kit version 3.0 from us. Unlike previous versions, he is unable to use it with his Microsoft Pascal compiler.

I understand that Andrew King at Microsoft is arranging an upgrade from his Microsoft Pascal to Microsoft C V6.0

Simon Lee
The Software Construction Company Ltd
Hertfordshire

EXE

An object lesson in programming

C++

Meet Your Objective

Object-Oriented Programming (OOP) is programming for the '90s. It's the next step after structured programming and is a much more productive way of writing applications. Borland has combined the power of OOP with the efficiency of C to produce new Turbo C++ Professional®.

Turbo C++ Professional is the first Turbo-charged native code C++ compiler that brings Object-Oriented Programming to your PC. Since Turbo C++ Professional also compiles ANSI C code, you can be productive with C now, and move to C++ at your own pace.

Environment ++

The best compiler deserves the best environment and our new Programmer's Platform® turbo charges your productivity. It features overlapping windows and mouse support, as well as a new multi-file editor, an integrated debugger, and a smart project manager. Its advanced open architecture lets you integrate the tools you need and are familiar with.

VROOMM adds room

VROOMM® (Virtual Run-time Object-Oriented Memory Manager) lets you break through the 640K barrier. Just select the application code you want to overlay, and VROOMM does the rest – swapping modules on demand. It's fast, easy, and automatic.

Another +

Turbo C++ Professional gives you all the tools you need to build fast, reliable C++ programs.

Turbo Debugger® 2.0 debugs your object-oriented programs. This powerful new version is the first and only debugger to support reverse execution, letting you step backwards through your code to find the bugs you might have missed.

New Turbo Profiler®, the world's first interactive profiler, gives detailed analysis of your program's performance. With it, you can easily spot execution bottlenecks and see where improvements or redesign of your code will yield maximum performance gains. And Turbo Assembler® 2.0 lets you replace time-critical segments of your code using the world's fastest MASM® compatible assembler.

Buy objectively

The suggested retail price for Turbo C++ Professional, including Turbo Debugger, Turbo Assembler and Turbo Profiler is £249.95 plus VAT. Turbo C++ costs £149.95 plus VAT. For further information complete the coupon and return or call our PRODUCT INFOLINE on (0628) 771070 or simply talk to your dealer.

Turbo C++ Compiler

- C++ conforming to AT&T's 2.0 specification
- C++ class libraries
- Full ANSI C compiler
- VROOMM overlay manager
- Complete documentation and tutorials

Programmer's Platform

- Open architecture for integration of your own tools
- Overlapping windows with mouse support
- Multi-file, macro-based editor
- Smart project manager provides visual MAKE
- Integrated debugging and hypertext help

YES! Please send me a free Turbo C++ demo disk. ❒ 3 1/2" or ❒ 5 1/4"

NAME: ______________________

POSITION: ______________________

COMPANY: ______________________

ADDRESS: ______________________

______________ POSTCODE: ______________

.EXE-10-90

TELEPHONE: ______________________

Please return this coupon to: Borland International (UK) Limited, FREEPOST RG1571, Twyford, Berks RG10 8BR

BORLAND

CIRCLE NO. 247

The Protection Racket

Software theft is rife. How best can you protect your program from the bandits? Willie Watts surveys the alternatives.

My first .EXE article to achieve the coveted status of Cover Feature was a survey of methods of anti-software pirate measures. When I was writing in 1988, protection systems (both software and hardware based) looked to be on the way out. Lotus had just abandoned protection on 1-2-3. August journals such as *Byte* and *PCW* regularly laid into any software house that dared to present an uncopyable program for review (reflecting, I suspect, the atypical priorities of the review journalist. If he has six different packages to compare, with mutually incompatible software protection, he is much more likely to see red than the punter-in-the-street, who only buys just one, and whose perception of the value of the software is actually increased by having a physical dongle).

Since the original article, things have moved on. The journalistic dislike of anti-pirate systems may remain, but protection systems flourish. The evidence: the pages of this Magazine, which previously contained about one or two adverts per issue, now abound with protection systems, to the point where a certain friend has nicknamed us '.DONGLE' magazine. My original article covered products distributed by just five companies - which was all that I could find at the time. In the preparation of *this* article, I contacted 16 manufacturers/distributors, and I am aware of at least one that I was obliged to leave out.

I remain in no doubt about the usefulness and validity of protection, including dongles (but see separate box for another view). The remaining question, *What sort of protection do I need?*, is more difficult, and is influenced by such factors as the retail value and function of your product, the likely ferocity of the attack and the target market. For example, dongles are very popular with companies exporting software. More surprisingly, some corporates actually prefer that suppliers dongle their software, as a measure to enforce company discipline.

Different types

Before considering disk manglers and dongles, a quick reminder of the computer-free methods of protection. If your software comes with a whopping great set of manuals, requires a massive wodge of technical support to set it up and keep it running, and you sell it on a site licence basis (or can afford to let customers behave as if you did), it may be that you are already home. Smaller manuals may be printed in dark brown ink on light brown paper: this not only gives them a fashionable, ecological look but also defeats photocopiers.

The next step up is probably software-based protection. This is generally considered less secure than a hardware device, but it is usually cheaper, and certainly less hassle. Software protection typically works by encrypting the .EXE file that you wish to protect. The decryption routines are usually tied into a unique (we hope) feature of the customer's machine; his hard disk 'fingerprint' (for example, a table of disk flaws), a floppy key disk, or even the timing characteristics of the machine's electronics. Because it is possible to store encrypted data relating to the program, software protection in this form can be very versatile. You can create demo versions which refuse to run after six goes, or after the Ides of March, but which spring to life when a password (exchanged for folding currency) is typed into the program.

Hic est dongle

From software-based protection, all roads lead to dongle. Just in case someone doesn't know, a dongle is a device which plugs into your computer. The protected software will only run if it is there. The dongle is convenient for the customer. He is free to make backups. To run your program on a different machine, he has merely to unplug the dongle in one place and plug it in elsewhere.

Modern dongles are mostly designed to fit on the parallel printer port, although some are available for serial ports and PC/XT bus compatible slots. These days, they are all transparent, ie it is possible to connect them between a computer and a printer without affecting communications. They are mostly cascadable (you can connect two or more to the same port without them interfering with each other), and, in practice, dongle conflict is a rare problem. Where it occurs, it can be solved by the purchase of a cheap printer port card. Dongles are reliable beasts - none of the users that I interviewed reported unacceptable failure rates or compatibility problems, although there was an occasional printer or lap-top computer (which often underpower their ports) that gave a little trouble.

Dongles function in many different ways. Simple dongles return a single, fixed code number. Some dongles return a stream of pseudo 'random' numbers, similar to that provided by high-level languages' random number generators. More sophisticated dongles yet allow you to 'seed' the random sequence, thus creating a dialogue between software and hardware. There are also dongles which encrypt data sent to them and, a popular innovation this, dongles which contain a few bytes of read/write memory. This memory can be used to store the same sort of information as encrypted data of software-based protection; for example, the particular configuration of your software.

Dongles are generally supplied tailored to your code or range of codes, so that nobody else's dongle can ever match yours. For some types of dongle you can acquire a customisation kit which lets you set up some of the internal codes, enabling you, for example, to mark dongles as 'demonstration' or 'full application'.

The manufacturers also offer different methods of linking the dongle to your software.

At worst, they tell you the necessary port numbers and leave you to get on with it. The best systems offer a program which encrypts .EXE files (as with software protection) **and** provide a suitable calling interface and object module for your development language.

What is available?

Faced with 16 distributors (some labelled with phrases like: 'We challenged the entire Maths and Computing department of Harvard University to crack our product; after three months they admitted defeat'), and given two weeks in which to review them *and* edit the magazine before you, I had to cheat. The comments on protection given above, and many of the observations that follow, were mostly supplied by real customers of the companies, who I have been calling up. Many thanks for their help. To keep this article down to size, I have concentrated on one product per distributor. The companies are listed in 'Leslie Halliwell' alphabetical order.

BL Security Ltd

Main Product: Deadlock I. Parallel port dongle, is accessed via 16 byte password, and contains 16 bytes of memory. Password and memory can be set up by user, using a special programming device (£155) - 8 reprogrammings allowed. It is not possible to alter the dongle's memory 'on the fly'. Programmer incorporates calls to dongle in his code; dongle is accessed via supplied standard object module - the latter is encrypted with data identifying the programmer device used to set it up, for extra security. About a dozen standard MS-DOS interfaces supplied.

Unit Price: 5-20 £16.50, 30-70 £13.75, 100-500 £9.75, 700+ £7.50. The supplier wishes it to be known that prices for the 'gaps' in the bands are negotiable. BL Security is also running a promotion until the end of 1990, where all orders for 50+ units are priced at £7.50 each.

Other information: This product and company are quite new, no user data available on quality of support. Dongle manufactured in Israel, originally developed for Israeli MOD. Other products in the range include a more sophisticated dongle with read/write memory, and a software utility that 'donglises' applications automatically. Existing users chose dongle because of good value for money, and small size of operation (seen as virtue because dongle less likely to have been cracked). Contact 081 343 0734.

Brent Communication ATC

Main Product: Maxpro. Parallel port dongle, with read/write memory. Dongle's memory may be programmed 'on the fly', and contains 6 bytes of general purpose memory, but much more formatted to support special features. Software supplied with package encrypts MS-DOS program file in the style of software-based copy protection; this system *must* be used - no language bindings are provided. As with some software-based copy protection, this means that you can't use overlays internal to the .EXE file (because the overlays will not be decrypted when loaded). Encryption program is very full-featured; for example it is possible to create 'evaluation' programs which cease to run after a date/number of runs/duration of running, but which the customer can re-enable by entering a password (supplied in exchange for his credit card number). The dongle also can detect attempts to hack it, and contains 'tamper links'. If one of these is broken, the unit will cease to operate and must be returned to the supplier for resetting.

HULTON-DEUTSCH

Unit Price: 1-49 £30, 50-99 £25, 100+ £20. Also required: a copy of the dongle-programming software - £65.

Other information: UK made - Maxpro was originally produced to meet an inhouse need. It is reputed to offer an unusually high level of security. The philosophy of Brent's auto-encrytion process is that Brent has had much more time to study the techniques of creating hacker traps, and so will be much better at it than the ordinary application programmer. Brent's sales approach is unusual: discovering that a certain company used another dongle (the advanced Softlok model, qv), Brent stated that it could break the protection. The challenge was accepted... and the software was cracked in a few hours. The victim subsequently switched to Maxpro.

Brent also offers a range of more 'conventional' dongles, both with and without memory, which may be accessed by drivers in the usual way. Contact 0423 566972.

Bristol Software Factory

Main Product: The Software Key, a serial port dongle, provides streams of pseudorandom numbers (seedable), which can be sampled to provide a password or used to encrypt protected data. Some example code to access the dongle is provided (GW-BASIC, Microsoft C for MS-DOS and OS/2, Turbo Pascal V3.0-V4.0), but the manual encourages the user to write his own - on the basis that the absence of a standard driver will create more difficulties for the hacker.

Unit Price: 20-99 £33, 100-499 £31.50, 500-999 £30, 1000+ £28.50

Other information: UK-made, customers include Racal and Hewlett-Packard. The dongle is unusually large (over 11 cm in length) - this has led to some complaints from end-users: 'I can't push my machine close enough to the wall' etc. The stream of random numbers provided by the hardware provides the potential for well-protected systems; but the onus is on the user to set it up - I heard two accounts of software protected with this dongle being hacked. The standard and quantity of documentation/software is perhaps a little disappointing; however existing users expressed satisfaction with the level of support offered. Contact 0272 629790.

Clearsoft

Main Product: Microphar Memory Key, a parallel port dongle, with 62 bytes of read/write memory. Access via encrypted object modules which must be linked to application. Vast assortment of high-level languages explicitly supported, including oddballs such as Metaware C, Baby 36 etc. Works under MS-DOS, SCO Xenix, P-System (UCSD Pascal), OS/2 and PROLOGUE (who?). Interrupt vector 1 & 3 blocking utility plus various other anti-debugger software supplied.

Unit Price: 10-29 £39, 30-49 £36, 50 - 99 £35, 100-199 £30, 200+ £28.

Other information: French made. Product range well-established on Continent, comparatively new to UK market. Customer reported delivery time: about two weeks. UK support was rated very good - for example, an extra component was supplied free to solve a particular problem. Some compatibility problems with laptops. No manual available yet - it is being written.

Company supplies several other dongles, including an 'all-purpose' serial port dongle, for use on workstations, minicomputers etc. Contact 091 378 9393.

Comsec

Main Product: KeyPro, a parallel port dongle, with about 10 bytes of read/write memory. Access via encrypted object modules which must be linked to application, or via automatic modification of .EXE file with KPINST. Example code supplied for four high-level languages (MS-DOS). Dongle has metal (shielding) case.

Unit Price: 1-10 £33, 11-25 £31.£35, 26-49 £29.70, 50-100 £28.05, 100-500 £26.40.

Other information: British made. Comsec is a well-established company, but the KeyPro product is a relatively recent innovation - so the customers that I interviewed had not had much experience with it. Comsec's support is rated highly. Order turnaround was 1-3 weeks.

Comsec has other dongles as well as Keypro and the serial port dongle, unusually including an internal PC bus dongle. Contact 0234 751203.

Control Telemetry Ltd

Main Product: Model 521, a serial port dongle, returning simple, fixed code. Access via link to non-encrypted object module, for which assembler source is provided, or automatic modification of .EXE file with Zareba program. A good selection of high-level language bindings, including some quite obscure offerings (including some rare-ish FORTRANs). MS-DOS only. Documentation is very extensive. A separate programmer is available (£186), to allow the software developer to customise (partially) the dongle's internal code.

Unit Price: 10-99 £40, 100+ £35.

Other information: British made. This dongle is virtually the grand-daddy of them all (manual makes references to Sirius/Victor machines etc). This means that the technology may be simple and long-in-the-tooth but, on the other hand, it has also been well-tested. One CTL customer commented that he thought that the 521 was probably overpriced, but he remained faithful on a 'better the devil you know' basis.

CTL has other dongles, including the recently-announced 527, which is a programmable parallel port unit. Contact 071 328 1155.

The Data Business

Main Product: Minder software-based protection system. Disk based system; user can operate software with distribution disk acting as key disk, or move protection to hard disk. Protection of individual programs is accomplished by running a utility which encrypts the .EXE file. Unusually, you need to buy each key disk (ie protected copy) individually; there is no 'infinite lives' version of the program. If you restrict the user to key disk style operation, you can link in a run-time module which may be called at intervals to verify the presence of the key - otherwise it is only checked once at start-up. Documentation was being revised at time of writing.

Unit Price: 10-90 key disks £6 to £8 (for 360 KB to 1.44 MB), 100+ key disks £3 to £4. 'Metered' disks are available for commercial duplicators, this would probably work out cheaper. One off purchase of basic utility (£99) is also necessary, the runtime module is extra (£29).

Other information: Minder has been in existence for five years, and has been distributed by various different companies. Originally owned by WayDisk, The Data Business acquired the copyright in July 1990. Level of security is not clear; one ex-user (who I rang up regarding another system) abandoned Minder some time ago after his application was hacked. However, revisions are issued from time to time, so the situation may have improved. The complicated pricing structure, based on add-on modules and copy counts, seems fussy and old-fashioned. Contact 0865 842224.

Data Encryption Systems Ltd

Main Product: Deskey DK12, a parallel port dongle, which provides a stream of pseudo-random numbers. Access automatic modification of .EXE file with DESlock program; or via user-written driver - BASIC and assembly language examples are provided. Documentation is clear but thin.

Unit Price: 10-24 £26, 25-99 £24, 100-499 £22, 500-999 £20, 1000+ £18

Other information: British made. Users report that the DK12 is much more robust and reliable than predecessors - a previous model was sensitive to the printer attached to it. Support rated highly. Order turnaround usually about two weeks, but can be one day in an emergency.

Serial and bus versions of the dongle are also available, these are also based on the random number principle. Contact 0278 653456.

Empiric Ltd

Main Product: MemoHASP, a parallel port dongle, with about 500 bytes of read/write memory. Access via automatic modification of .EXE file with HASPINST program and linkable object module (manufacturer recommends both applied). Fair selection of high-level language bindings, plus 'generic' TSR driver for use by unsupported languages. Additional 'Pattern Code Security' feature is included to make the application code harder to hack. Documentation is very professional.

Unit Price: 1-9 £45, 10-24 £40.50, 25-49 £39.37, 50-74 £38.25, 75-99 £37.12, 100-150 £36, 150+ £33.75

Other information: Users report variation in performance of units, ie sometimes a given unit will fail to work with one particular machine. Turn-around on orders was less than one week.

Empiric also supplies a non-programmable version of this dongle. Contact 0628 35913.

Link Computers

Main Product: Cop's Copylock II software-based protection system. Protection can operate either as a 'key-disk' system or installed onto a hard disk. In the latter case, the program works by measuring certain aspects of the host machine: tolerances of motherboard components, floppy disk controllers etc. This enables user to backup, reformat hard disk and restore software without needing protection moving facilities. However, if he replaces the HDC, he's stuffed. Copylock is added to programs by modification of .EXE files - no programming is required. Facilities are provided for making Demo versions (with limited number of runs), automatically generating Serial Numbers etc. The documentation is excellent.

Unit Price: Standard version £600 (key disk only), Automatic version £1200 (installable on machine). One-off payment, allows the protection of as many disks as you like.

Other information: Danish product. Lotus is evaluating the system. User reports suggest that Copylock is very secure. However, two individuals complained about its sensitivity to hardware. Apparently, it refuses to install on some machines, especially portables. Once in place, the protection sometimes fails after a few minutes, when the machine warms up, or after a few weeks, when the characteristics of the machine change with age. The program can be 'tuned' to be less sensitive... in which case the protection breaks down between machines of the same model. Another complaint is that there is no method of deinstallation, to allow an application to be moved between machines. Contact 0763 263 073.

Magnifeye

Main Product: Hardlock E-Y-E, a parallel port dongle, contains a sophisticated encryption algorithm - potentially very secure. Access via automatic modification of .EXE file with HL-Crypt program and linkable object module (manufacturer recommends both applied). About 16 high-level language bindings supplied. Documentation and presentation the best of any product described here, by about one order of magnitude. A separate programmer card is available (price £50) which allows customisation of dongles by software house.

Unit Price: 1-49 £34, 50-99 £31, 100-199 £28, 200-499 £26

Other information: German made. Very high quality product. Dongle is very small (less than 5 cm long) - ideal for lap-tops. Users report delivery times of less than one week. The 'E-Y-E' in its name refers to a transparent panel on top of the unit, through which the ASIC chip on which the dongle is based can be seen.

The dongle is also available with a 128 byte memory, and as an XT bus card. Contact 071 221 8024.

Microcosm

Main Product: CopyControl software-based protection system. Disk-based MS-DOS system. Protection is achieved by automatic modification of the .EXE file by the CCADD utility, or by linking in and calling an object module. CopyControl is very full-featured, offering the ability to create demo versions (software stops working after a number of runs, or after a given date), remote upgrades (by supplying a password), choice of permitted number of working copies, usable movable protection and so on. A LAN version (£120 extra) allows the restriction of the number of simultaneous users on a network.

Unit Price: £3 per disk + £50 for the initial kit. An 'infinite copies' version is available for £419.

Other information: British product. This is the current choice of one user, who has struggled with Minder and Cop's Copylock (both qv). His only remaining complaint is that it is not yet possible to inspect the status of the copy protection on a returned disk - but Microcosm is working on it.

For more details of the protection system, see Peter Cheesewright's article on page 28 in this issue. Contact 0272 441230.

Rainbow Technologies

Main Product: SentinelPro, a parallel port dongle, contains an encryption algorithm. Access via a linked object module. About 35 high-level language bindings supplied, with support for MS-DOS, XENIX, UNIX and OS/2. Documentation is good.

Unit Price: 10-24 £38, 25-49 £34, 50-99 £30, 100-499 £25, 500-999 £22

Other information: US made. Rainbow Technologies used to have a poor reputation for order turn-around times, but this has been cured now that the company arrived in the UK. Rainbow is popular with the giant software houses - for example, Autodesk, Digital Research and Microsoft - perhaps because of its large size and stability. Uniquely, Rainbow included its company report with the dongle review kit.

Rainbow has a large range of other dongles, including the recently introduced Sentinel-Scribe, which contains 120 bytes of memory. Contact 0753 41512.

Software Security

Main Product: Activator/M, a parallel port dongle, contains 128 bytes read/write mem-

BRIEF

ory. Access is via user-written code; examples are provided in C and Turbo Pascal. Manual is poor (very hard to understand) - but I am told that a rewrite is underway. Customisation of the dongles is possible on purchase of a separate field programmer - but see note below.

Unit Price: 1-9 £40, 10-49 £36, 50-99 £31, 100-499 £26, 500-999 £24, 1000+ £22.

Other information: US made. Recent introduction to the UK market. Order turnaround from US reported as 3 - 7 days. One user had purchased a field programmer six months ago, but had been unable to get it to work despite a stream of replacement parts sent out to him. However, dongles performed satisfactorily, and the Activator is well established in the US. The ancillary materials with this dongle (manual, example programs) were significantly below par.

Software Security also offers the Activator, which is identical except that it has no memory. Contact 0784 430060.

But What Happens When?....

I can't say that I like dongles very much. I appreciate the commercial justifications for using dongles. What I'm not so sure about is that they are as foolproof and safe to use as their manufacturers would have you believe.

Dongles are plugged into a parallel or serial interface port on a computer, and software routines installed into a protected program engage in a dialogue with the dongle to ensure that it is present and correct. The dongle can be checked whenever the program's author likes: on start-up, or frequently whilst the program operates.

Dongles are presented as the acceptable face of copy protection. They don't diddle with your floppy disks or hard disks, and they don't stop you taking backup copies of your programs and data. You can install as many copies of the program as you like on different computers: you can carry the dongle around with you from machine to machine. The dongle stops you using the software on more than one machine at a time.

The September 1990 issue of .EXE contained ten adverts for dongles (it also contained two adverts for 'pure' software-based copy protection systems). Either .EXE readers buy a lot of dongles already, or the various advertisers have a very good reason for thinking that .EXE readers will buy a lot of dongles in the not too distant future.

So what happens when a dongle-protected applications program only checks the dongle on start-up? One dongle can be shared between several pirated copies of the software as long as they are started up at different times.

Users often want to use more than one software package on their machines. So what happens then, when each package uses a dongle? If the dongles are compatible, each one can be plugged into another (making a nice little chain of devices to hang off the back of the computer). If the dongles are not compatible, the user has to swap them back and forth as each piece of software is used. This gives rise to 'dongle shuffle', a repetitive plugging and unplugging, in which something (usually one of the gang-banged dongles) breaks.

What happens when you are using a multi-tasking package like DESQview and you want to run two dongle-protected packages at once? If the dongles are incompatible, you can't do it.

What happens if your portable PC doesn't have a parallel port that fits your dongle? You can't use the protected software.

Dongles successfully avoid some of the biggest criticisms of copy protection techniques. You can copy dongle-protected software as many times as you like to take floppy disk and tape backups; dongle-protected software can be installed on networks and servers (as long as the dongle checking routines are compatible with the network software); there is nothing to stop you re-installing should you reconfigure your system and the need for data backups is not compromised. However, if you have a dongle-protected software package, you won't get a backup dongle (since dongles are very reliable in practice, giving away backup dongles would mean that software producers were giving away an extra copy of their software).

In which case, what happens when you lose your dongle, or when you break it? (see 'dongle-shuffle' above or when someone pushes your PC against the wall behind your desk), you'll have to wait until you can get a replacement. If your dongle is tied to your software's licence number, you'll either have to wait for a new dongle to be programmed or you'll possibly have to re-install your software when you get the replacement dongle. Either way, you'll be in for a wait and some work.

What happens when you use dongle-protected software for a critical application? If something went wrong with the dongle and I was using a £25,000 software package, I would want to have at least two backup dongles!

Maybe there is some sense for using dongles with high-end, high-cost vertical market programs where a pirated copy could mean a significant loss of income for the software producer. In such cases users will need support which means that pirates will sooner or later call on the authors: if there are pirated copies about, the authors will hear about them. Then you can play the 'What happens when...?' game in the courts.

Paul G Smith

Softlok International Ltd

Main Product: SOFTLoK, a parallel port dongle supplying a fixed code response. Access is via linked object module (supplied with source) or directly from high-level language. About a dozen high-level language bindings provided (MS-DOS). Documentation minimal but clear.

Unit Price: 10-99 £12, 100-499 £8, 500+ POA

Other information: British made. Comparatively low level of security (hardware dismissed as 'trivial' by one ex-user), but price extremely competitive. Technical support rated highly. In case of SOFTLoK dongle conflicting with another dongle on the same port, the company has in the past supplied a printer port card for £20 (it has promised to do this for free in the future).

The company also offers SOFTLoK PLUS, a memory dongle. It is currently developing Softlok II, a memory dongle to be launched the first quarter of 1991, at about 20% above the price of SOFTLoK. Contact 0254 772220.

User Friendly

Main Product: Everlock software-based protection system - disk-based MS-DOS protection, achieved either by automatic modification of .EXE files with EVBUILD, or by linking object module and interrogating protection (or both). Product very full-featured, with facilities like permitted backups, moving protection to/from hard disks, serial numbers and so on. New features with latest version include support for Windows 3.0 and prevention of concurrent use on networks. Documentation is good.

Unit Price: 125 user counts starter pack £195, unlimited edition £465.

Other information: US made. Like Microcosm's CopyControl, is vulnerable to users who reformat their hard disks without first 'moving' protection back onto floppies. One complaint made by user; from the customer's point of view, there is no way of finding out where protection is, short of trying to run protected application.

User Friendly offers a mass disk duplication service, and can supply Everlock software drivers for use in various models of mass duplicators. The company also offers Evertrak, a security system which encrypts the program file with user name/serial number - any attempt to remove it stops program working (£200). A dongle is under development. Contact 0527 585550.

EXE

DES File Encryption

You need a license to export a program containing the DES algorithm - despite the fact that it is in the public domain. Tony Miller explains one of the most popular encryption algorithms.

Figure 1 - Substitution table

Plain text:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Cipher text:	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A

Confidential files held on your PC are vulnerable to inspection. Even when you are careful to lock your machine, or change your password regularly, or keep your office locked, there is always the chance of breakdown. Once the machine is out of your hands, with the lid off, then anything on the hard disk is accessible.

If your particular machine is used by others, or if it's a networked system with security holes, the problem is the same. Even UNIX is rumoured to have defects!

Where is the computer security problem worst? Is it the boardrooms of the top 100 companies? No! Is it in the Whitehall corridors of power? No! The worst case scenario is in a typical University department of electronic engineering and computing. Teams of dedicated and spasmodically brilliant students can spend three years attacking your computer security. They may know that exam questions (and answers) are held on computers, but that is not the aim. The real challenge is to defeat the system. It is my job to see that they don't.

Encryption

The only way to guarantee the security of files held on a computer hard disk is to encrypt them. No 'ifs or buts', it has to be done as a matter of routine for all critical files. The best ciphering (encryption and decryption) algorithm for general use is the

Where is the computer security problem worst? In a typical University computing department

Data Encryption Standard (DES), published by the US National Bureau of Standards. It is said to be unbreakable (...but see later). There are other methods for transfer over networks, or for special applications, but for general use for files on a hard disk, or on a floppy disk sent through the post or even for Email, DES is the winner.

To understand how it works, let's look first at the basic principles of ciphering. If you have had occasion to open one of the standard text books on cryptography, you may remark that they look like test pieces for the latest in desk-top publishing, with a prize for selecting the most bizarre font. It happens that much of the formal notation and mathematical techniques of cryptography are similar to those in areas of information theory and signal processing. Indeed, one of the key papers in the field was written in 1949 by C E Shannon, a name well-known to electronic engineers.

Since I am writing this with one of IBM's best pieces of software (the Personal Editor) I will skip the hieroglyphics.

Encryption is the transformation of a clear file into an encrypted file, using a particular key. Decryption is the reverse process, using the same key. The algorithm used must be reversible, and it is up to the user to keep a secure record of the key. There are other ways of arranging matters, but this is the one of practical significance for file security.

The clear file is made up from a character set. Practical algorithms just transform each character in the clear file into another character in the encrypted file. This implies that the two files are of the same length, ie we have a one-to-one mapping of the clear file (or plain text) into the encrypted file (or cipher text). The DES uses the binary character set, but, to make it easier to follow the principles of encryption, my examples are confined to the 26 characters of the alphabet.

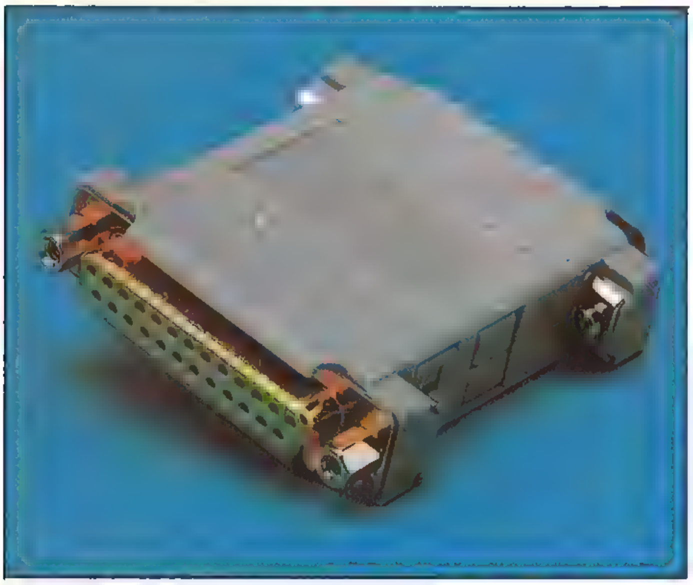
There are just two ways of changing text while still preserving a one to one mapping. The first is transposition: rearranging the text to change its sequence. For example, UNIX becomes XINU. The second is substitution, that is replacing each letter with another. For example, UNIX becomes VOJY (I hope that isn't anyone's trademark) by incrementing each character. You can

Plain text:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Cipher text:	S	E	A	M	B	C	D	F	G	H	I	J	K	L	N	O	P	Q	R	T	U	V	W	X	Y	Z

Figure 2 - Substitution table with keyword

SECURITY UPDATE

SOFTLoK & SOFTLoK PLUS
software protection systems



How many users of your software actually paid for it???
Our SOFTLoK range of software protection devices are used by hundreds of software developers from small consultancies to large multinationals. They are designed and manufactured by us in the UK.

SOFTLoK
SOFTLoK units contain a code that can be read and verified by routines provided in assembler and a variety of high level languages.

Price: 10-99 units £12.00, 100+ units £8.00
Evaluation kit £20.00

SOFTLoK PLUS
SOFTLoK PLUS units contain a small area of non-volatile memory protected from unauthorised reading or writing by a password. Both the memory area and the password can be changed at any time by the routines provided in assembler and several high level languages. As the memory can even be altered in the field it can contain counters & expiry dates as well as serial numbers etc.

Price: 10-49 units £40.00, 50+ units £35.00
Evaluation kit £40.00

- ⋆ For IBM PC, PS/2 and compatibles
- ⋆ Installed in seconds in the parallel printer port
- ⋆ Assembler & high level source code provided
- ⋆ Fast response
- ⋆ Transparent to printer
- ⋆ Small physical size

SOFTLoK INTERNATIONAL LIMITED was established in 1987 with the introduction of our SOFTLoK and SOFTLoK PLUS devices to help combat the ever increasing problem of software piracy.

SOFTLoK has now broadened it's range of products to tackle unauthorised access to PC's which is an increasing threat to the business community.

SOFTLoK has a wealth of experience in the computer security market place. As well as supplying our standard products we have also designed and manufactured customised dongles for several Blue Chip companies.

If you are concerned about your computer security and would like to talk to us further, or you would like to place an order, ring us now on (0254) 772220.

. . . . As PC's become indispensable to your business, the data they contain becomes critical to it's survival. In many cases that information is your business.

DISKLoK II
PC security system

It is easy to forget that the unprotected computer sitting in your office may be insured under your contents policy but the data on that PC could well be priceless.

DISKLoK II provides complete security for your office PC and also your laptop's out in the field. DISKLoK is easy to use for the computer manager and transparent to the computer user.

- ⋆ Up to 100 users
- ⋆ Boot limit protection
- ⋆ Virus detection
- ⋆ Auto screen blanking
- ⋆ Directory/File encryption
- ⋆ For Standalone PC's
- ⋆ Audit trail
- ⋆ Floppy/DOS restrictions
- ⋆ Secure menu maker
- ⋆ Protects existing directories
- ⋆ Installed in minutes
- ⋆ Uses about 2k of memory

Prices per copy
1-9 £149, 10-49 £99, 50-99 £89, 100 POA

All prices excl VAT and delivery

SOFTLoK
INTERNATIONAL LIMITED

Units 10 & 11, Railway Road Industrial Estate, Darwen, Lancashire, BB3 3EH, England
Tel (0254) 772220 Fax (0254) 705956

CIRCLE NO. 252

write out a little table to help the ciphering (see Figure 1).

To make the algorithm stronger - more resistant to the bad guys who are trying to break your code - the first move is to use a key. For a transposition cipher, the plain text might first be broken down into blocks of eight characters, thus

`Have a nice day.`

becomes

`/Have a n/ice day./`

Each block of characters is then permuted according to a key thus:

Plain text block: `/Have a n/`
Key: `6 5 7 0 2 4 3 1`
Cipher text block: `/en aaHv/`

Here the first character in the plain text becomes the sixth character in the cipher text, the second character becomes the fifth and so on. This is repeated for each block of eight characters. The permutation is reversed for decryption.

In a substitution cipher, the keyword is first rewritten so that no characters are repeated, for instance, SESAME becomes SEAM. A table for ciphering is then written out with the reduced keyword first, followed by the rest of the alphabet, but excluding the letters in the keyword - see Figure 2. To avoid the obvious overlap towards the end of the alphabet, the keyword may be inserted later in the alphabet. Thus if the keyword has four letters, as in this case, we might adopt the solution shown in Figure 3.

Figure 3 - Substitution table with shifted keyword

Plain text:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Cipher text:	W	X	Y	Z	S	E	A	M	B	C	D	F	G	H	I	J	K	L	N	O	P	Q	R	T	U	V

If we can afford the luxury of a key which is at least as long as the plain text and if the key will be used only once, then an unbreakable algorithm can be produced. The characters in the alphabet can be represented by the numbers 0 to 25. Each character in the plain text is then added to the corresponding character in the key to produce the cipher text, eg

Plain text: `unix`
Key:`abba`
Cipher text: `uojx`

This is unbreakable, because there is only one key that can transform the plain to the cipher text and all keys are equally likely. However, if you happen to be a reader of this magazine who also watches old Eurovision Song Contests it is possible that you might guess it. Such people are unlikely to exist.

This system was said to have been used during the Second World War. Agents were given a pad of key texts. Home base held a duplicate. Each message used one sheet of the pad for encryption and was then disposed of, hence the name one-time pad. Unfortunately, the possession of such a pad was incriminating.

However, the principle is very important to modern cryptography. The method of combining the plain text and the key sequence is not critical to the security of the algorithm. A pseudo-random key sequence can be generated and this forms the basis of a class of systems called stream ciphers. Having whetted your appetite, I shall say no more about them and concentrate on block ciphers, the other principal class of cipher.

Strength of algorithms

Transposition does not change the frequency of occurrence of a given letter in the text. This makes it vulnerable to attack with methods based on the frequency of use of each letter in the language ('e' occurs most often in English). It does destroy the sequence of letter groups. The high probability of a letter b being followed by a letter i cannot be made use of. Substitution has the opposite effect, changing the frequency of occurrence of letters but not the relationships between them.

In 1977, the hard men at IBM came out the winners in a contest to produce a data encryption standard for the US National Bureau of Standards. The DES is an open standard and all details of the form of the algorithm are published. Some aspects of the assessment of its security remain classified. We are assured by the US National Security Agency that the DES has no known statistical or mathematical weakness.

An analysis of the strength of an algorithm is a complex business, but the conclusions are simple: use a long key and combine

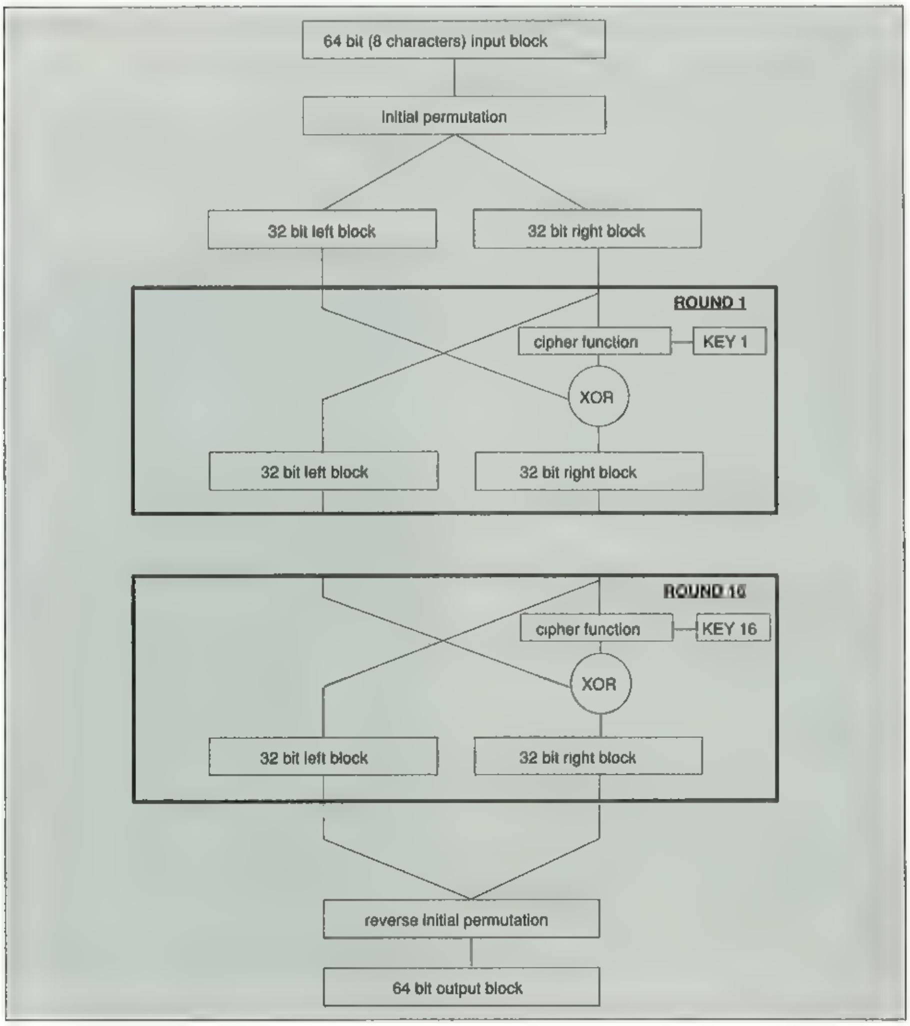


Figure 4 - Encryption algorithm

transposition and substitution in multiple rounds, ensuring that the value of each character in the encrypted file depends upon all the other characters in the input block and in the key. So, the DES uses a ciphering function with both substitution and transposition, with 16 rounds of ciphering. Each round uses a different key, which is derived from the original key. This gives the ease of use of an 8 character key with some of the security features of a longer key.

The Algorithm

The DES is a block cipher, working with 8 characters at a time. It was designed for hardware implementation, so it works with the ASCII representation of the 8 characters as a block of 64 bits. The principles involved in transposition and substitution described earlier for characters are equally applicable for bits. The working set of 26 characters is simply reduced to a set of two, 0 and 1. The overall enciphering algorithm is shown in Figure 4.

The first move is to generate the set of 16 keys. Characters are in 7-bit ASCII. This allows the 64 bits of the 8 character key to be reduced to 56 significant bits and a permuted choice made to split the 56 bits into two 28-bit registers. A sequence of left shifts on each half followed by a permuted choice generates each 48-bit key in turn.

The plain text file is dealt with 8 characters at a time. The 56 significant bits of the block of 64 bits are first split into two blocks of 32, using a fixed initial permutation. A copy of the right block is taken for use as the left block in the next round of the algorithm. The right block is then encrypted with the ciphering function and XORd with the left block to form the right block for the next round of the algorithm. When dealing with binary coded data the XOR is a particularly useful function since, to decrypt, all you need do is repeat it.

The Cipher Function

The ciphering function is shown in Figure 5. The right block (32 bits) is first expanded via a permutation to 48-bits. This clearly involves using some of the bits more than once. The resulting 48-bit block is then XORd with the current key (48-bits). Then comes the tricky stage, to reduce the 48-bit block back to 32 bits without throwing away any information (we will need to decrypt it eventually!). This is done using a set of devices called S-boxes.

The 48-bit block is first divided into 8 blocks, each of 6 bits. Each 6-bit block is fed to a separate S-box, which reduces it to 4 bits. Figure 6 shows the layout of one S-box, with the numbers in decimal notation. It forms a look-up table. The first and last bits of the 6-bit input block give the row number (0 to 3). Bits 1 to 4 give the column number. The numbers looked up in the S-box convert to 4-bit numbers in binary notation. The eight 4-bit blocks are then combined to give the required 32-bit block.

Text books on cryptography look like test pieces for the latest in desk-top publishing

Since the number selected from the S-box depends on all the bits in the 6-bit block the process will be reversible on decryption. A further permutation is then applied before the block is XORd with the left block as shown in Figure 4.

There are 16 rounds of encipherment with the 16 different keys. Finally, the inverse of the initial permutation is applied to give the output block.

SID, the Software

OK, so how do we do it in software? The program design for my shot at the problem, SID (Software Implemented DES), follows the specification of the algorithm pretty closely. It has a simple user interface, consisting of command line entry in the form

```
SID keyword filename
```

If an incorrect format is used, or the keyword does not consist of 8 characters, the correct mode of entry is described. To preserve security, the program will write the encrypted file over the clear file. Since this is potentially disastrous, the first few lines of the file are displayed on the screen and the user must make a positive decision (by pressing the Y key) to continue with the encryption.

If the decision is to continue, the list of 16 keys is generated from the keyword with a function `gen_keys(char *keyword, int *keylist)`. This first converts the 8 character keyword into a 64-bit integer array. The array is then divided into right and left 28-bit arrays to follow the key generation algorithm outlined earlier. The look-up table specified by the DES is modified slightly, without affecting the result, to allow the two blocks to be permuted directly into the keylist in the form of a 16 by 48-bit array.

The input file is then reopened for binary read and write and the function `crypt_it(int *keys)` used to en-

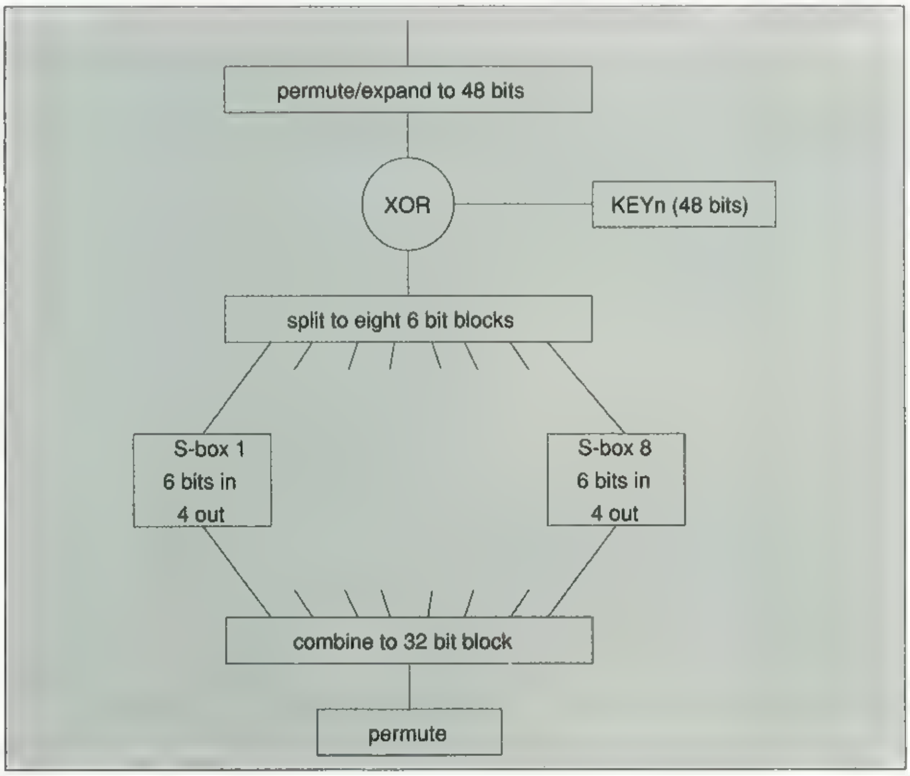


Figure 5 - Cipher function

Figure 6 - S-box 1

Row	Column 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

crypt the file, 8 characters at a time. The various bitwise manipulations can be handled directly, since the blocks are just held in integer arrays with array element values of 0 or 1. The binary/decimal conversions needed for the S-boxes are handled with look-up tables. The final conversion of bit arrays to ASCII codes is also handled with a look-up table.

It is important not to leave any files around which could be accessed using toolkits or otherwise. It is also worth checking that the encryption has proceeded satisfactorily before overwriting the original file. SID first loads the encrypted file into a temporary file. It then checks that the original and temporary files are of the same length. If they are, it copies the encrypted file on top of the original file, overwrites the temporary file with garbage and then explicitly deletes the temporary file. If it finds a problem (none so far!) it deletes the temporary file, leaves the original unchanged and reports the problem to the user. With the Microsoft C compiler, SID will encrypt or decrypt at around 1 KB/sec on a 386 PC.

To decrypt, all you need to do is repeat the process, but this time with the keys in reverse order. Everything else will sort itself out. I have called my decrypting program 'DIS'. The user just enters

```
DIS keyword filename
```

Since the algorithm is published, there is no need to keep it secret. Keyword security is of course vital, and it is often necessary to educate users on the choice and safekeeping of keywords. The important operational peculiarity of file security, compared to other forms, is that if a file has been accessed you may not know it, whereas if, say, a safe has been blown open it is usually noticeable!

Breaking the Cipher

Fine, you might say, but I don't rate the US National Bureau of Standards, or even IBM, very high on altruism. Would they really release an algorithm that didn't have loopholes in it? The DES may be fine for University use, but how about serious commercial security? This may be a problem, particularly as recent newspaper reports suggest that, in the current climate of Superpower détente, the CIA is devoting more of its resources to commercial espionage. I guess the other side is as well.

In 1977, the hard men at IBM came out the winners in a contest to produce a data encryption standard

Two main objections have been made to the strength of the DES algorithm. The first is to the design of the S-boxes. It has been suggested that this is not random, which raises the possibility of a quick method of solution via a trapdoor in the algorithm.

The second objection is to the relatively short key, effectively eight 7-bit characters (56 bits). This increases the likelihood of obtaining a solution by trying all possible keys. The cost of a highly parallel hardware engine to do this, and achieve a solution within 24 hours, has been estimated at several million pounds. Only the largest organisations are likely to be able to afford this. However, supposing that you want to rule out this possibility, what can you do?

The first and simplest option is to encrypt more than once, using different passwords each time. This should defeat any exhaustive search technique. The second option is to change the algorithm. Simple options include increasing the number of rounds from 16 to, say, 32, or changing the numbers in the S-boxes or elsewhere. With only slight additional programming effort, the length of the keyword could be increased, or 8-bit characters allowed. A fundamental disadvantage of this approach is that it may then become necessary to keep the algorithm itself secure. To avoid this, a self-modifying algorithm, transparent to the user, could be used. For example, the number of rounds used could be a function of the keyword.

Any more substantial changes should be considered cautiously. The underlying theory of the DES is subtle.

EXE

Tony Miller started using computers with the CEGB in the days when you had to call someone in from head office to write your software. He is now a senior lecturer in the School of Electronic and Electrical Engineering at the University of Birmingham. He is also school computing manager and head of the IT Research Group, specialising in security and the user interface. He takes every opportunity to climb the most remote hills these islands can offer.

If you are interested in reading further, Dr Miller recommends the following books: Cryptography *(by Carl H Meyer and Stephen M Matyas, pub John Wiley, ISBN 0-471-04892-5) is the standard and relatively readable work by the IBM experts;* Cipher systems *(by Henry Beker and Fred Piper, pub Northwood Books, ISBN 7198-2611 X or 7198-2571-7 in paperback) has lots of exercises and is complete with lemmas, theorems and corollaries;* An Introduction to Cryptology *(by Henk C A van Tilborg, pub Kluwer Academic, ISBN 0-89838-271-8) is a recent book with a good introduction to number theory, but heavy going.*

If you would like a copy of the software described in this article, please send a blank, formatted disk and a stamped addressed envelope, as described in the 'Editorial' notes on Page 1. Please follow the instructions exactly or we may not be able to return your disk. Mark your envelope 'DES '.

SID has been submitted to the University of Lancaster NPDSA.

Beyond Copy Protection

When Peter Cheesewright set out to design a copy protection system, he faced a multitude of design decisions. We asked him to explain his choices.

Being the UK distributor of a third party copy protection system, and additionally being software developers ourselves, we were sensitive to the protection requirements of software developers. The third party package did not meet all their requirements, so we set about producing an 'ideal' system from scratch.

Our spec stated that the adopted system must work on all IBM compatible machines running any version of MS-DOS, DR DOS etc. Furthermore, it must allow the software to be run from any standard disk format (hard or floppy), and be compatible with any applications or utilities that might be used.

Copy protection systems work by locking the software to a unique attribute on the computer system. We considered all the various possibilities.

We rejected the approach which relies on finding a unique feature of the machine itself. Two 'identical' machines from the same manufacturer are hard to distinguish from each other. Variations, within certain tolerances, of electronic components could be used - but the components may vary more with temperature than between machines. Reliability could be a problem.

Some machines have battery backed-up RAM which could possibly be used to hide a unique serial number... but many machines do not have this feature. No soap.

Dongles are an easy way of achieving good levels of security, although a dongle system is only as secure as the software that interrogates the dongle. This did not bother us - software was our forté. We eventually rejected dongles, however, on the grounds of cost (especially prohibitive for demo versions), compatibility (conflict when 'daisy-chaining' different dongles) and user acceptability (we felt that we had detected an adverse user-reaction to dongles).

Then there was the 'key-disk'. A 'key-disk' is very much like a dongle in principle. It does, however, have the big advantage of costing nothing extra. It has various disadvantages. It prevents/makes it more difficult to use the floppy drive for normal purposes, it advertises when it is being accessed (and so is easier to break), it is even less acceptable to the user than a dongle and it wears out.

We concluded that the only part of a microcomputer that varied from machine to machine in a stable, measurable fashion was the hard disk drive. The disk controllers were also potentially uniquely identifiable, but varied too much between PCs to be used as a reliable method.

Disks can readily be altered by software, so we considered adding a unique 'fingerprint'. However, in order to remain totally compatable with everything, we wanted to avoid doing anything non-standard to hard disks (more about floppy disks later). Fortunately, we found a way of identifying ostensibly identical hard disks, without adding any special sectors or tracks or any other non-standard feature. You will understand that I cannot tell you any more, without blowing my own product!

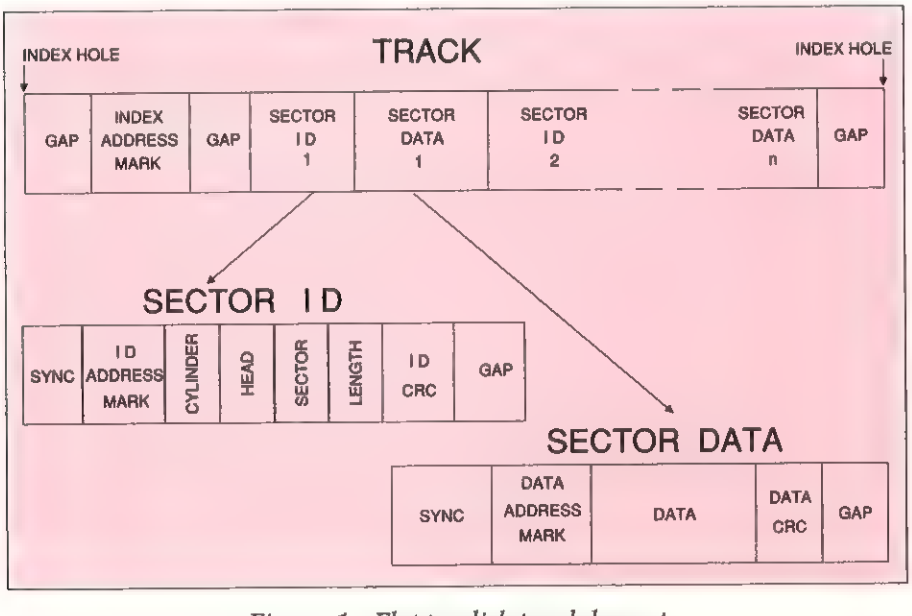


Figure 1 - Floppy disk track layout.

Floppy disk protection

Software is distributed on floppy disk. We had to ensure that the software was securely locked to a particular floppy disk until it was installed. We were up against 'Bit-copier' products, such as CopyIIPC and CopyWrite, which had been specifically designed to break the protection on floppy disks.

The simplest method of protecting a floppy disk is to add extra tracks at the end of the disk. For example, a standard 360 KB floppy has 40 tracks numbered from 0 to 39. Most copy programs stop at track 39, or when they meet an unformatted track. These copy programs can be beaten by leaving track 40 unformatted and adding track 41. Unfortunately, some recent machines will not access tracks beyond number 39.

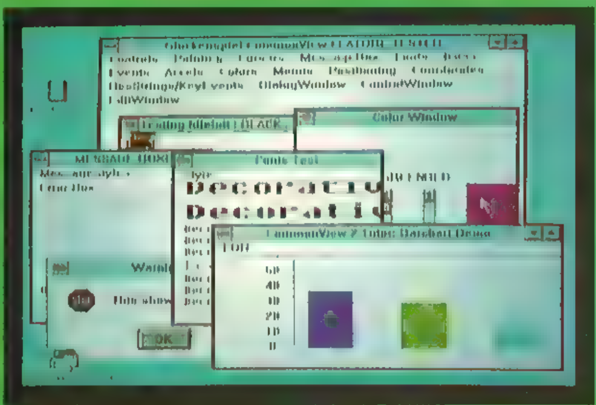
Objects of Desire

glockenspiel
CommonView 2

***Glockenspiel CommonView** has really made its mark in the world of Windows development. Thousands of developers have used it to speed up and simplify their projects.*

Now **CommonView 2** is available, inheriting the success of its predecessor and extending its capabilities even further to deliver efficient Windows 3.0 apps.

That's because **CommonView 2** works with **Glockenspiel C++ 2.0**, giving you a C++ object-based framework that reduces the complexity, cuts the code, manages memory and lets you stay in touch with what you're really doing. From compilation to execution, **CommonView 2** applications are fast and powerful.



CommonView 2 objects model the real world of Windows. And that's just the beginning - from there **CommonView 2** takes you as far as you want to go.

It doesn't anticipate everything. That's where C++ takes over: its inheritance capabilities enable you to extend the **CommonView 2** framework. Just take an existing object and customize it without affecting the original. A few additional lines of code and you could, for example, turn an edit control into a password control. Then reuse it in other applications.

You *don't* have to learn a complete new language, you *can* integrate existing developments; it *is* portable to PM and NewWave; CodeView *can* be used on your C++ source.

The world wants Windows 3.0 apps now! And with **CommonView 2** you can inherit the earth.

Specifications.

Glockenspiel CommonView 2 includes Glockenspiel C++ 2.0 and Container - the object storage framework. It requires Microsoft C 6.0, the Windows SDK and 1.5 meg of memory. You debug C++ source with Microsoft CodeView 3.0. Glockenspiel C++ supports a completely portable memory management system. Glockenspiel CommonView consists of approximately 65 classes.

Comprehensive documentation includes CommonView tutorial and reference manual, Glockenspiel C++ compiler manual and User Guide, C++ syntax and AT&T Library Guide, pullout guide to compiler switches, *plus* "Programming in C++" by Stephen C. Dewhurst and Kathy T. Stark (Prentice Hall).

On-line hypertext documentation for CommonView reference manual and AT&T guides. Tutorial source code also on disk.

Glockenspiel C++ works from within the Programmer's Workbench with the reference guides instantly available from the on-line Advisor, using Microsoft CodeView for debugging.

Glockenspiel CommonView applications are portable between Windows 2.1 and Windows 3.0, PM 1.1 and PM 1.2 with HP New Wave 3.0 version coming soon.

glockenspiel
class constructors

QA

QA Training Limited,
Cecily Hill Castle,
Cirencester, GL7 2EF.
Phone: 0285 655888. Fax: 0285 650537.

CommonView costs £495 + VAT and is available for Windows 3, OS/2 Presentation Manager and HP NewWave.

Glockenspiel Professional C++ 2.0 Compilers are available for DOS, OS/2 and Workstation platforms. Call for details.

CIRCLE NO. 255

PANEL® *Plus* II

Screen Generator and User Interface Library for 'C'

PANEL *Plus* II *features:* Interactive screen design editor • C and MS-FORTRAN code generators • library of user-interface functions • pop-up fields/windows • multi-line fields • mouse support • scrolling regions/ scroll bars • help boxes • pull-down menus • custom validation • shadow borders • complete **library source**, with low-level source and headers for all supported systems (DOS, OS/2, Unix, VMS) • interfaces to popular PC **graphics libraries** • OS/2 DLL • **no royalties.**

Also available: **Utility Source Option** which allows **PANEL *Plus* II** to be ported to any suitable development platform. *New in version 2.1:* **CUA/SAA** compatible key assignments and edit functions, CUA 'function key area' support, word-wrap editing, new menu styles, **MS-C 6.0** interface (with on-line help), DOS-extender interfaces.

Panel *Plus* II licence fee per programmer: **£295** (including library source and 700-page manual). With utility source option: £720. Please call for quantity pricing, full version list, and free demo disk.

Roundhill Computer Systems Limited

Orchard House, Ogbourne St. George, Marlborough, Wiltshire SN8 1SU
Telephone: **0672 84 535** • Fax: 0672 84 525

Roundhill also supplies compilers and development tools from Copia International, Graphic Software Systems, Lattice, Periscope, Pocket Soft, Polytron/Sage and others. Please call for a full catalogue and price list.

CIRCLE NO. 256

A floppy disk is formatted as a number of concentric tracks each containing a number of sectors. The start of the track is located by the disk drive by means of a physical index hole. At the start of each track there is a header followed by a number of sectors (see Figure 1). Each of these sectors has a header, some data and a tail. Normally, the data part is all that DOS has access to. Most copy protection systems play around with the non-data parts of the track.

One method of protecting a disk is to introduce 'bad' sectors by, for example, modifying the checksum (CRC) at the end of the sector data. An ordinary copy program will either give up or write the data to the new disk with a corrected checksum. Either way, the new disk will not have the bad checksum. The copy protection software can detect this and know that it is an illegal copy. 'Bit-copiers' are not fooled by this scheme and will readily reproduce bad sectors.

One system physically removes part of the magnetic media in order to produce bad sectors. To check the disk, it writes to the affected area and then tries to read the data back. On the original disk, the result will be unpredictable. On a copy, it will read something sensible and, therefore, know it is invalid. The disadvantage of this method is that it is sensitive to disk alignment and requires the use of special disks.

Instead of creating bad sectors, it is possible to change the sector numbering scheme. Each sector on a track is preceded by a sector number (see Figure 1). Changing this number can fool certain disk copying programs. Similarly the number and size of sectors on a track can fool some copiers. One trick is to put so many sector headers on the disk that the copying program, thinking that each is 512 bytes long, runs out of buffer space.

It is possible to produce 'weak' bits on the track. These generally occur when a write is made with the write head turned off. Instead of being definitely '0's or '1's they are somewhere in between. It is very difficult for a copy program to reproduce these weak bits reliably.

Other schemes require the use of special hardware to put the protection on the floppy. We rejected this approach, partly because, as things turned out, it proved to be unnecessary, but also because it would make it impossible to transfer protection legally to other floppy disks.

Although only the data parts of a track are normally accessible, it is possible, through assorted trickery, to change virtually any part of the bit image on a track. We use such techniques to implement a combination of changes that is hard to beat either with special software or hardware. The combination is important. Although it may be possible to beat one technique alone, in so doing it makes it difficult to beat another one on the same track. If you then vary the protection method from disk to disk, you end up with what we have - a very secure system (although I say so myself) using reliable techniques that are independent of disk speeds and other timing considerations.

EXE

Peter Cheesewright is Managing Director of Microcosm Ltd, a small software house formed 10 years ago. Its claims to fame so far are the world's first RAM-disk for micros (Silicon Disk) and the world's first disk cache for micros (MicroCache).

The product referred to in this article is CopyControl. It is one of the systems described in the Protection Racket article, elsewhere in this issue. CopyControl is available directly from Microcosm (0272 441230).

An Orange Book

What does it mean when an operating system is advertised as 'secure to level C2'? Bob Mitze is the man who knows his oranges.

With the expansion in demand for Open Systems in general, there is a proliferation of networked systems. Such systems are often easily accessible to the public; perhaps even being connected to phone lines. This, in turn, increases the need for high standards of system security. Various companies are now looking to produce tested, secure systems which provide universally accepted standards of security for Open Systems. It is in this climate that reference to the 'Orange Book' is becoming more and more frequent.

The Orange Book, properly called the *Trusted Computer Systems Evaluation Criteria*, derives its name from a bright orange cover. It is published by the US Department of Defense and provides guidelines for developers and buyers of trusted systems. It was originally intended for those operating in the realm of US Government and military contracts. Commercial implications have arisen because the Orange Book standards have become more widely accepted and acknowledged, and companies have started to brand their products as compliant with one or another level. This has considerable market impact, since gaining compliance above the C level - discussed below - is a difficult and lengthy process.

Seven levels

The Orange Book sets out seven, well defined levels of security. These levels are grouped into four bands, namely A (Verified Protection), which is the highest level; B (Mandatory Access Control); C (Discretionary Access Control) and D (Minimal Security), with subdivisions within B and C.

To satisfy class D requirements all you have to do is fail to meet the criteria demanded by the higher grades - so no problem there!

The next level in the hierarchy is C1, (Discretionary Security) the level at which most commercial multi-user systems are categorised. A named user of such a system must be able to prevent his files being read or destroyed by another user. In effect, a user would have to prove his identity, in all likelihood with a password. The security system itself must be secure against tampering, and the system in its entirety formally tested and documented. UNIX System V, for example, provides most of this as standard, so extra kits are almost certainly not needed at this level.

C2 relates to Controlled Access. Access to a file here can be permitted or refused down to the level of any one particular user. No-one can ever be allowed to see another user's discarded memory or disk space even if the data in it was already encoded. A record must be kept of all login, opening, creating or deleting events and the system administrator's actions must likewise be audited.

When you move up to the MAC tier, the lowest level is B1 (Labelled Security), moving up through B2 (Modular Structure) to B3 (Isolated domains). Mandatory Access Control is not standard in UNIX systems, so a kit will probably be required to achieve this level. MAC can be rather restrictive. Every object in the system is given a sensitivity label, not unlike the notions of 'Top Secret', 'Secret' and so on down to 'Unclassified'. Each object or file has only one sensitivity level and is accessible only to those users who are cleared up to that particular level.

Files are also categorised according to their contents (for example: 'salaries', or 'sales' or 'accounts'). A user is restricted to entering only those files which relate to his specific area of concern and only then, of course, at the level to which he has been cleared. All hard copy printed out from the system would automatically be labelled with its sensitivity level and category information, and dealt with accordingly. Changes to an individual's clearance level and category membership would be strictly controlled since, obviously, opportunities for self-promotion would defeat the system.

At the very top of the scale, Verified Protection, there is only one category - A1 (Formal proof of correctness). You cannot 'bolt on' security features into a product to achieve A1 classification. It has to be designed from scratch to comply. You must provide 'a formal model of security policy... including a mathematical proof that the model is consistent with its axioms'. Heavy stuff. I cannot conceive of a situation where a commercial organisation would request a supplier to develop a system to the A1 secure standard - this is the point at which military and commercial applications part company.

While the Orange Book defines very precise levels of security, it should be appreciated that no computer system is really secure unless sensible standards of administration and application are set up and followed. It is also important to evaluate very carefully which level of security will best suit your requirements. It is a waste of both time and money to provide too high a level of security. That subtle combination of acceptable levels of cost, restricted use and suitable security is one you must work out for yourself - the Orange Book will not do it for you!

EXE

Bob Mitze is the Managing Director of UNIX System Laboratories, Europe (USLE) - formerly AT&T UNIX Software Operation Europe, Ltd. The company, which is a wholly owned subsidiary of AT&T, has responsibility for the sales and marketing of UNIX System V and associated products throughout Europe. UNIX is a registered trademark of UNIX System Laboratories Inc.

The Orange Book, DoD 5200.28-STD, 'Department of Defense Trusted Computer System Evaluation Criteria' is available from: Office of Standards and Products, National Computer Security Center, Fort Meade, MD 20755-6000, Attention: Chief, Computer Security Standards.

Hidden horrors

Something wicked this way comes... and it's invisible!
Fridrik Skulason reports on a class of MS-DOS virus that is especially hard to catch.

Good camouflage improves the chances of penetrating enemy territory without being noticed. The development of 'radar invisible' aircraft by the United States Air Force is testament to the role of deception in warfare. The tag *stealth* has now been purloined from the military to describe a series of computer viruses which attempt to hide themselves from 'the enemy' - ie the combined forces of users and anti-virus software. Specifically, 'stealth' refers to the group of viruses which make the virus code disappear from the infected media.

The original stealth virus was the Brain virus, which overwrote disk boot sectors. Once loaded into memory, the virus would intercept every BIOS Int 13h operation. If an attempt was made to read the boot sector, the virus would instead return the contents of the original non-infected boot sector. The virus could be detected in memory, but while it was active, all infected diskettes appeared clean. This simple method is expected to be a common feature of future boot sector viruses.

In the case of 'parasitic' viruses - ie viruses which 'parasitize' program files - the implementation of camouflage methods is more complex. Two conditions must be met:

- No increase in file size is detected when the user issues a DIR command.
- A program reading from the file must not see the virus code, only the original contents of the program.

One type of virus, the so-called 'companion' virus, fulfils both these conditions - yet it is not clear whether to include it in the stealth category. Companion viruses exploit the fact that if two programs exist with the same name, but different extensions (.COM and .EXE), MS-DOS will execute the .COM file. These viruses work by locating a .EXE file and opening a new file in the same directory, with the same name but a .COM extension. The virus code is placed in the new file, which is hidden by setting attributes. When the user attempts to execute the .EXE file, the .COM file is activated instead. The virus does whatever it was designed to do, then runs the .EXE file. TPworm and AIDS II are examples of companion viruses.

Conventional stealth viruses do indeed alter program files. The approach adopted by the 'Number of the Beast' virus is of considerable interest, as the virus code is hidden in unused, free space after the end of the program itself. The method has one serious drawback (from the virus writer's point of view) - the virus code is not included when the DOS COPY command is used to copy an infected program.

The most advanced method for hiding any increase in file length consists of intercepting the 'Find first' and 'Find next' functions of DOS Int 21h. If the information returned indicates that the file is infected, the virus modifies it, returning the original length of the program. This method is used by Zero Bug and 4K (Frodo).

If a virus based on this method is active in memory, it is possible to damage programs irreparably with CHKDSK. This is because a mismatch will occur between the number of clusters in use according to the FAT, and the length of the file 'disinformation' supplied by the virus. Running the DOS CHKDSK program may result in a number of reported errors in the FAT. If the user attempts to correct this, by running `CHKDSK/F`, the virus-occupied clusters will be freed, making it impossible to recover the original program.

Condition 2

The second condition of 'stealthness' is more difficult to implement. Two methods currently used for hiding virus code are described here. More sophisticated methods are known to exist, but they have not been found in existing viruses and so, for obvious reasons, are not given.

Method 1 involves intercepting 'open file' function calls and determining if the file being opened is an infected program. If so, the file is disinfected before control is passed on to DOS. Any anti-virus program opening a file for examination will not detect virus activity.

A virus using this method, 4K for example, can be removed by a very simple disinfection method. Entering the command

```
COPY *.* NUL
```

in each directory will remove the virus from all infected programs. However, this clean-up technique will fail, of course, if the virus re-infects the file as soon as it is closed, so that files are 'clean' whenever a program examines them, but infected otherwise.

The second method involves intercepting the 'Read' function. When a part of the virus code is read, the contents of the input buffer is replaced with the original code. This method is used by the 'Number of the Beast' virus, which overwrites the first 512 bytes of infected files. When this part of the program is read, the virus locates and returns the 'correct' contents of the first 512 bytes.

One of the most common anti-virus measures, a checksum program, is vulnerable to stealth type viruses. Such programs assume that the program file read from the disk is identical to the program which will be executed. The importance of running anti-virus programs *only* after the computer has been booted from a known 'clean', write-protected system diskette must, therefore, be strongly emphasised. Be *careful* out there!

EXE

Fridrik Skulason is an acknowledged virus authority and Technical Editor of the quality monthly journal Virus Bulletin. *He is currently studying for an MSc at the University of Iceland, Reykjavik.*

This article has been reprinted, with permission, from the September 1990 issue of Virus Bulletin. The annual subscription to VB is £195. If you are interested in finding out more about VB, please phone the publisher on 0235 555139, or write to Virus Bulletin Ltd, 21 The Quadrant, Abingdon Science Park, Oxon.

Tandon launch the fastest 486 computer under £5,000.

Which is more mind bending: the speed of our new PCAsl/486 or the £4,999 price tag?

Let's look at the speed first. For starters, there's an Intel 80486 processor that crunches numbers like they're going out of fashion.

(Tests have indicated that the 486sl runs at a break-neck 10.3 m.i.p.s.)

Just as hair-raising is the 486's built-in maths co-processor (it tots up figures faster than a bookie on Derby Day). And our unique Power Poster feature, a high speed cacheing system, sprints into action so quickly it'll bring tears to your eyes.

However, £4,999 doesn't only buy you sheer unadulterated speed.

There's a cavernous 110MB of hard disk storage capacity and the high clarity VGA colour monitor makes the most humble pie charts look tasty.

And don't forget that the 486sl can use not only existing PC compatible software, but also the latest 32-bit graphical interfaces such as Windows 3.

Take heed though, at under £5,000 we predict the 486sls won't be hanging around for long.

For more information on the Tandon PCAsl/486 write to Tandon plc, FREEPOST, Hunt End, Redditch, Worcester B97 5XP. Or call us on (0527) 550550.

Tandon

Name .... *Position* .... EXE 3.90/486

Company .... *Address* ....

Telephone .... *Number of employees* .... *Number of PC's installed* ....

Prices exclude VAT 80486 is a registered trademark of the Intel Corporation Windows 3 is a registered trademark of the Microsoft Corporation

CIRCLE NO. 260

Zortech's Revenge

Zortech's monopoly on native code-generating MS-DOS C++ compilers ended with the appearance of Turbo C++. Paul Smith reviews the upgrade that must outface the competition: Zortech C++ V2.1.

Last March we reviewed V2.06 of Zortech's C++ development system. In July, Zortech shipped an upgrade to its product. This article describes how the new V2.13 of Zortech's C++ differs from V2.06. I looked at the 'Developers' Edition' of the product, which includes the Library sources, Debugger, and C++ Tools packages.

The changes

Zortech C++ 2.1 isn't an implementation of AT&T's C++ 2.1 specification. It is, however, a major upgrade to the Zortech C++ 2.0 product. Enhancements fall into several areas, notably the compiler, the debugger, a new development environment, and the 'C++ Tools' class libraries.

The most important enhancements are directed towards making good use of the limited memory available for DOS applications. Zortech supplies versions of the compiler and debugger that use Rational Systems DOS Extender technology to run in extended memory on 268, 386 and 486 systems. Up to 16 MB of memory can be used by DOS Extended applications, as long as you have sufficient extended memory available. This innovation allows large C++ applications from linearly-addressed memory systems to be compiled on MS-DOS computers.

Users of Zortech C++ 2.1 can use the Rational Systems DOS Extender technology in their own programs if they wish, as long as they purchase a copy of the Rational Systems development kit.

Fighting back against Borland, which has given users of its C++ compiler access to its VROOMM technology, Zortech has implemented its own Virtual Code Management (VCM) system. Like VROOMM, VCM splits programs into small segments, which are loaded and unloaded dynamically as and when required. VCM requires no source code changes, and is implemented in the compiler and linker. Programs that use VCM can contain as much as 4 MB of program code.

The documentation has been extended mostly by means of add-on manuals that specify how the new products differ from the old (although 'C++ Tools' gets a whole new manual). Upgrading users will not be greatly troubled by this: new users could well be seriously confused. I would have preferred it if Zortech could have folded all the changes into one consistent set of documentation, although it is clear that the company has taken this approach in order to get the new version out as quickly as possible. Maybe we can look forward to further improvements to the documentation in due course.

Installation

The upgrade to V2.1 of the Developers' Edition of Zortech's C++ product comes with one new manual that describes the installation process and most of the differences between V2.1 and V2.0. There is also a supplementary upgrade manual that describes some more differences, a replacement for the 'C++ Tools' manual and a small booklet that lists a range of compatible development tools/products and also contains a further update to the Debugger manual. The software comes on 10 diskettes (5.25 inch version). The diskettes contain a number of READ.ME files that list yet more differences between the documentation and the product you have purchased. If you are buying Zortech C++ for the first time, you will receive compiler, standard library function, and debugger manuals.

As with V2.0 of the product, installation proceeds under the control of a program called ZTCSETUP. As promised, Zortech has answered the complaint in my review of V2.0: the program explains what it means before asking you whether you want it to write directly to the screen (as opposed to using the BIOS). However, I still think that the question is unnecessary.

ZTCSETUP isn't very good at upgrading an exisung installation. If it finds that Zortech C++ is already installed, ZTCSETUP asks if you want to overwrite it. It then asks the same question every time it finds that a new file has the same name as an old file. This is extremely boring: like me, you'll probably resort to deleting all the old files before you restart the installation process. You didn't store any of your own project files inside the Zortech directory structure, did you?

After you have installed the various files, you will (unless you are upgrading) need to make some changes to your system's AUTOEXEC.BAT file. Unfortunately, this part of the manual contains some errors.

Compiler enhancements

As well as providing DOS Extender protected mode versions of the compiler and related tools (see above), Zortech has made a number of other enhancements to the compiler.

There are two new memory models: the 'r' model, for compiling programs to use the Rational Systems DOS Extender, and the 'V' model for compiling VCM programs. Zortech still doesn't support the the 'huge' model, which means that if you want to port huge-model software from Microsoft C you will have extra work to do.

Pointers to members are now implemented, using a different technique to that employed by AT&T's CFront. Zortech claims that its approach is more efficient than AT&T's.

"The MKS Toolkit is an amazingly faithful replication of a System V UNIX™ environment."

– UNIX Review

Essential UNIX Tools for any PC user

"The entire MKS documentation package proved excellent in every respect"

Daniel McAuliffe, IEEE Computer, Jan. 1990

We can tell you that MKS Toolkit offers both experts and novices the purest form of UNIX utilities in a DOS or OS/2 environment. Fortunately, we don't have to. Software reviewers, universities and major corporations all over the world are discovering MKS Software and how it can help their programmers and software developers make the most of their creative talents.

With our proprietary code, MKS Toolkit offers you more than 150 UNIX System V. Release 4 compatible tools for DOS or OS/2. With MKS Toolkit, your computer or clone becomes a comfortable environment for shells, string matching, editing, file manipulation, and more.

In this one package you will receive a complete:

- MKS Vi full – screen editor
- MKS Korn Shell – user programmable interface
- MKS AWK – programming and prototyping language
- profiler – to analyse the execution time of code
- egrep/fgrep – fast pattern matching
- archivers, sort/merge, encryption, data compression & much more

Over 150 commands that help you get the most productivity out of your PC.

Easy Installation

With MKS's menu-driven installation you can be up and running in less than 10 minutes, no matter what your level of computer expertise.

"MKS software is absolutely the best in its class. Don't mix environments without it."

Grover Righter, Novell

Interconnectivity & Lan Licences

MKS Toolkit interacts well on most standard PC and PS/2 networks. Combined with Novell Netware™, the most popular LAN for PC's, MKS Toolkit creates a UNIX time sharing system in DOS or OS/2 organizations. UNIX workgroups can now hook up all of their PCs using PC-NFS™ and MKS Toolkit, enabling you to use a PC as a UNIX workstation and off-load your mini or main machine.

Spanning Novice to Expert

This versatile software reflects it's users' needs. UNIX experts will have all the familiar power they demand while novices will find our documentation filled with excellent tutorial material.

Organizations such as AT&T, Hewlett-Packard, ITT, and NCR – all heavily committed to the UNIX system – use MKS Toolkit to create a standard operating environment. Universities from King's College to the University of Berlin use MKS Toolkit to enrich personal research computing environments and double the potential of their PC teaching labs.

"With this package, you can become familiar with the UNIX environment on your microcomputer, with DOS only a keystroke away."

Byte Magazine, May 1989

Cost-Effective Learning Tool

If your organization is committed to moving into the UNIX environment, then MKS Toolkit is the perfect learning path. DOS or OS/2 users retain the familiar world of their PC keyboard and programs yet can move to a desktop UNIX environment at a keystroke. Exposure to new commands and functionality now becomes an integral part of the novice's working day.

MKS Programming Platform

MKS Toolkit can be purchased as an individual product or you can buy MKS Programming Platform which contains MKS Toolkit and three other MKS products. MKS Platform helps smooth out the details of programming and software development by adding time-saving utilities such as: MKS RCS (Revision Control system), MKS Make (an automated program builder), MKS LEX & YACC (compiler learning and construction tools).

In all, you simply cannot find a more complete set of commands and utilities to get from DOS or OS/2 to UNIX. Once you have used MKS Toolkit, programming will never be the same again.

US List Price

MKS Toolkit (DOS & OS/2)	$399.00
MKS Toolkit (DOS/2 only)	$349.00
MKS Toolkit (DOS only)	$249.00

Please contact MKS dealers for UK domestic prices.

The Software Construction Co. Ltd. (0763) 244114; fax (0763) 244025
Grey Matter (0364) 53499
Systems Science (071) 833-1022; fax (071) 837-6411

MKS

35 King Street North, Waterloo, Ontario, Canada N2J 2W9

Put The Power Of UNIX In Your PC.

CIRCLE NO. 261

The compiler is faster, as well: Zortech claims a speed improvement of up to 20%, although I wasn't able to make objective comparisons, because I had deleted V2.0 of the compiler from my hard disk. Code efficiency has also been improved. There is no longer any overhead for multiple inheritance unless it is used. The space used by virtual tables has been reduced, and, if you only use single inheritance, virtual calls are faster. That having been said, the performance of the compiler on the Plum Hall C benchmarks, as reported in my review of V2.0, is unchanged.

It is now easier to write dynamically linked libraries and programs for Microsoft Windows. Furthermore, Zortech have improved their product's ANSI C compatibility as well as making numerous other minor changes.

Virtual Code Management

VCM works by splitting a program into up to 255 segments, each of which can be loaded into memory on demand. The VCM software uses a least recently used algorithm to unload segments when there is no longer room for them. Segments are loaded only when they are needed, and remain in memory until they are discarded. VCM works on any PC, and doesn't require anything more than an 8088 or 8086.

Compiling a VCM program requires the use of the new 'V' memory model. All the modules linked into a VCM program must be recompiled for this model: that includes third party libraries. You don't have to make any changes to C++ or C source code, although you will have to modify assembler routines that make far calls to other functions. The segmentation is specified in the compiler or linker command line.

All calls between segments are routed through the VCM software, which loads segments into memory if they are not already present before routing the call into the segment. VCM is for real-mode DOS applications: it cannot be used in programs that are to run under Windows (which has its own segmentation system) or the Rational Systems DOS Extender (which removes the need for segmentation).

It allows large C++ applications from linearly-addressed systems to be compiled under MS-DOS

The Workbench

Zortech has replaced the ZED editor (the 'integrated environment') with a new Zortech Programmers' Workbench, or ZWB for short. ZWB does a better job of integrating the components of their product than did its predecessor. ZWB is a multi-window editor, that can open many files at once. It supports a mouse and uses pull-down menus. The compiler, linker, make, the debugger and the compiled program can all be invoked from within ZWB.

ZWB includes context-sensitive on-line help, a GREP-based browser, support for movable and resizable windows, copy and paste via a clipboard, and supports EMACS style keybindings and meta menus.

ZWB is extremely powerful and a vast improvement on its predecessor - but in its own way it is as idiosyncratic as was ZED. Although you can open many files at once in ZWB, each in its own movable and resizable window, you can't actually invoke ZWB from the command line with an instruction to open lots of files at once. If you specify a list of files, explicitly or with a wildcard, ZWB opens the first in a window, and then allows you to cycle through the list: however, each file is opened in the same window.

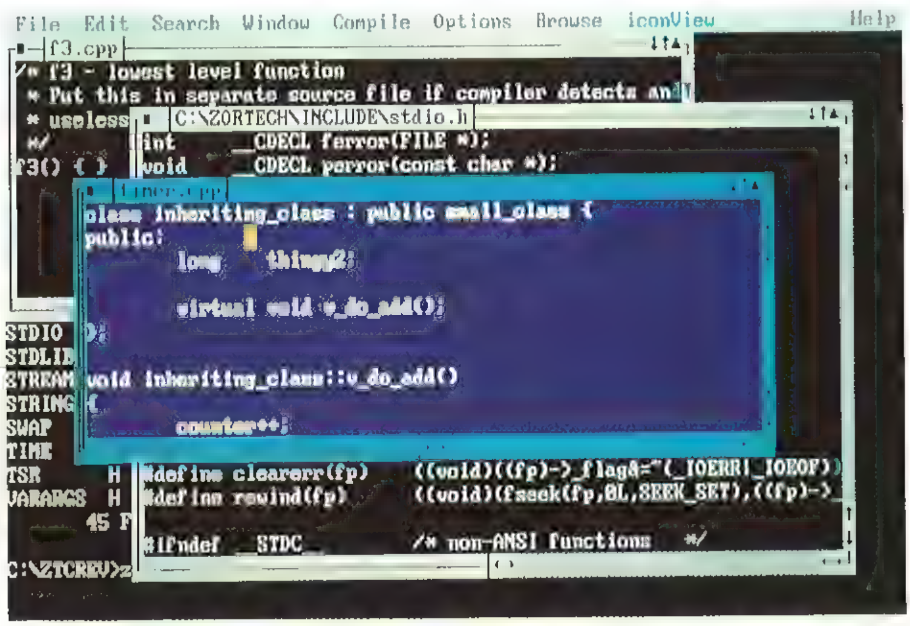
If you want to open lots of files at once, you have to use the menu commands and type in the file names one by one. Furthermore (as is admitted in the READ.ME file about ZWB), the mouse and menu handling is inadequate, and the menus are not operated quite the way one would expect. They still fail to provide a fully SAA/CUA conformant user interface. I also found that if I had set the command line for my program using the appropriate ZWB command, it remembered the first program's name and parameters even if I later started working on a new project in a different directory.

The browser is very useful: you can use it to GREP lists of files, and it can also be used to browse the definitions of C++ classes. However, don't expect it to be a Smalltalk-like class browser.

I must commend Zortech for its efforts with ZWB, which is certainly another great step forward. However, I hope that it can make some more improvements such as: allow multiple files to be opened from the command line; implement 'undo'; provide a directory file list picker so that file names don't have to be typed in every time; offer a real class browser; and make the user interface more SAA compliant.

Debugger enhancements

Zortech offers four types of debugger within the C++ V2.1 package. There is a normal debugger, like the old ZDB. There is a remote debugger, that links the slave up to a second PC through its serial port. There is a DOS Extended debugger that uses the Rational Systems extender and runs in pro-

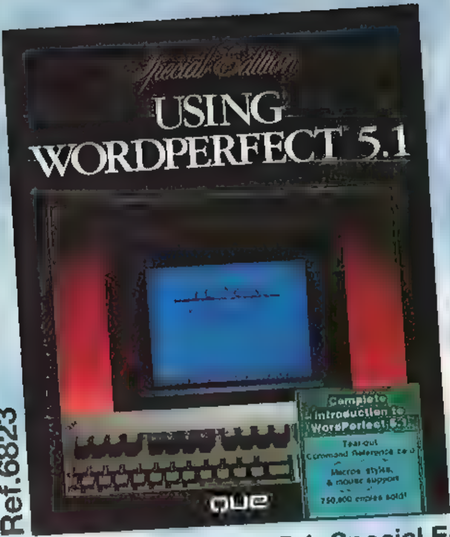


ADVANCED BUSINESS GRAPHICS

Figure 1 - ZwB is Zortech's new editor/environment

BEST SELLING QUE®

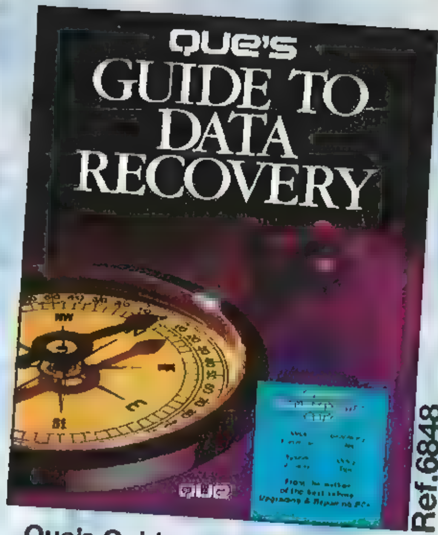
£22.95



Ref.6823

Using WordPerfect 5.1, Special Edition

Leads readers from WordPerfect basics to sophisticated techniques. Covers the latest software enhancements – including tables, pull-down menus, mouse support, "hot-linking" to spreadsheets and the new Equations Editor.



Ref.6848

Que's Guide to Data Recovery

Explains how users can retrieve lost or damaged data, implant system security, and prevent and recover from computer viruses. A variety of utility programs are introduced.

£27.45

Ref.6853

UNIX Programmer's Reference

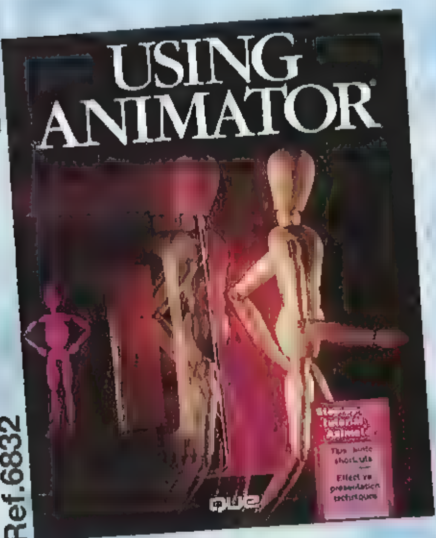
This advanced-level text is a vital resource for anyone who programs in the sophisticated UNIX environment. Filled with practical examples to help experienced programmers develop powerful skills and improve their efficiency. Essential for anyone who writes UNIX programs!

Ref.6828

SmartWare Tips, Tricks and Traps, 2nd Edition

Power techniques for SmartWare users! This practical guide contains hundreds of hints for SmartWare's versatile integrated applications. Includes timesaving techniques for Project Processing.

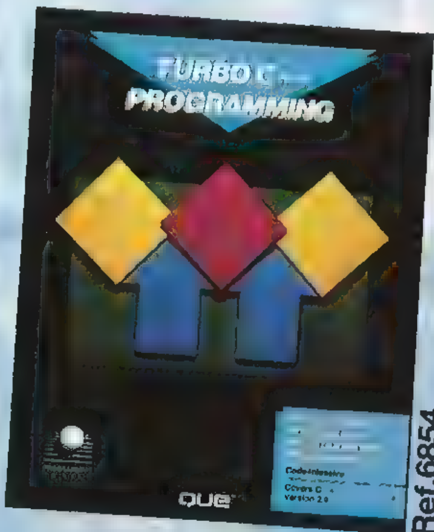
£22.95



Ref.6832

Using Animator

Complete, step-by-step introduction to animation graphics on a PC! This easy-to-use book explains Animator's versatile graphics capabilities and how to combine different effects to create a full-featured animated display. Includes a glossary of common animation terms and commands.



Ref.6854

Turbo C++ Programming

This advanced-level text introduces object-orientated programming for the C++ language. Filled with techniques to help programmers develop more powerful programs.

BOOKS MEAN BUSINESS

AVAILABLE NOW AT ALL GOOD BOOKSHOPS AND COMPUTER STORES

CIRCLE NO. 264

DISTRIBUTED BY: **COMPUTER BOOKSHOPS LTD.** 30 LINCOLN ROAD OLTON BIRMINGHAM B27 6PA **TEL: 021-706 1250**

tected mode on 286, 386, and 486 systems. Finally, there is a 'virtual 8086' debugger that takes over all extended memory on 386 and 486 processors and works by creating a virtual 8086 world in which the program to be debugged is run.

The three new debuggers offer various advantages to those that can run them, not least a reduction in the memory overhead. For instance, the remote debugger uses only 4 KB memory on the slave PC. The DOS Extender debugger has a 32 KB to 64 KB memory overhead and can coexist with other software that uses extended memory; the Virtual 8086 debugger uses only 4 KB to 8 KB of system memory, although it cannot coexist with other users of extended memory.

The debugger's user interface is not wholly consistent with that of ZWB. Particularly annoying is the lack of scroll bars in ZDB. Zortech insists that this is to distinguish between the editable windows in ZWB and the non-editable windows in ZDB. Rubbish! I still want to scroll around text in the debugger: why should I have to use a different technique?

New Tools

A full set of libraries is supplied with the product, including standard C and C++ libraries. Zortech still doesn't support the C++ V2.0 model streams library, which is a shame. As before, the Developers' Edition of Zortech's C++ V2.1 includes source code for all the libraries except Zortech's Flash Graphics package. The library source code is also available separately.

Zortech has made a lot of revisions to the 'C++ Tools' collection of C++ classes, which is supplied as a part of the Developers' Edition and is also available separately. The list of classes is now extended with: interrupt handlers, screen area classes, interactive and mouse-driven objects, buttons, menus, scrollable text, help panels. These classes can be used to implement a textual user interface with mouse control and menus.

Conclusion

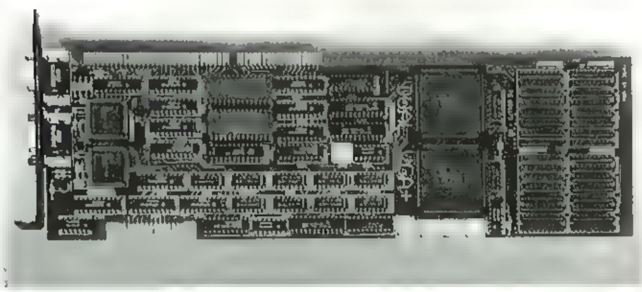
Zortech has improved an already fine product, and has made changes that will serve to answer those critics who compare it with competing products. Although I have a few small reservations about the ZWB programming environment, they are not strong enough to dissuade me from recommending Zortech C++ 2.1. If you want to develop C++ software for PCs, you should take a very close look at this product.

EXE

Paul G Smith is a freelance writer and software development consultant, who is presently juggling the release of a major new communications product, a consultancy contract, his writing commitments, and a hectic out-of-work schedule. He can be contacted on CIX and AppleLink as 'pgsmith'.

Readers may recall that at the end of our review of Borland's Turbo C++ (.EXE July '90), we printed a few comments on the new Borland product by Zortech's Steve Teale. We undertook to redress this piece of deliberate imbalance by inviting a Borland representative to comment on the next Zortech C++ release.

The invitation was duly made. However, after initial promising noises, the company regrettably chose not to participate in the exercise.



Amazing Multicolour Library Part II

Graeme Webster concludes his presentation of a library of code to drive the exotic 256-colour modes of the Super-VGA graphics cards.

In last month's episode, I gave an overview of the most popular types of Super-VGA graphics cards, and presented the code necessary to switch these cards into their special 256-colour modes. I also dealt with the concept of memory bank switching, explaining why it is an inevitable consequence of the PC architecture, and how to do it with the particular hardware under consideration.

As it stands, then, the library consists of `InitGraphics256()`, which switches any card into its 256 colour graphics mode, `EndGraphics256()` which returns it to text mode, and `_SetBank256`, an internal routine to control memory bank switching. You will note that, despite many hundreds of lines of coding effort, we have yet to dirty the screen. The first task, then, must be to get access to individual pixels.

Pixel Addressing

Fundamental to all drawing operations, and intimately associated with bank switching, is the calculation of pixel addresses. In all the video modes we are considering, there is exactly one byte per pixel, which at least saves the fiddly bit-shifting necessary in less colourful modes. Taking the origin at the top left hand corner of the screen, the byte in the video memory corresponding to the pixel at location *x,y* is:

*x+y *pixels_per_row*

```
; File name:  PAdrG256.asm
; Function:   Determine buffer address
; of pixel in 256 colour modes
; Caller MASM:    AX = y-coordinate
;                 BX = x-coordinate

; Returns: BX = BYTE offset in buffer
;          ES = video buffer segment

; sets banks, as necessary, via DX
VideoBufferSeg EQU   0A000h

_TEXT SEGMENT  BYTE PUBLIC 'CODE'
      ASSUME   cs: TEXT

      EXTRN _SetBank256:FAR

      PUBLIC   _PixelAddr256
_PixelAddr256  PROC  FAR


      mov   cx,ax      ; y now in CX

      push  ds
      mov   ax,seg _DATA
      mov   ds,ax        ; DS -> DATA
      mov   ax,_PIXELS_PER_LINE
      mul   cx
; lsb in AX, msb in DX
      add   ax,bx
      jnc   L1
      inc   dx          ; handle carry

L1:   push  ax  ; offset in AX
      call  _SetBank256
; get offset in BX to match routines
; in Microsoft 'Video Systems'
      pop   bx

      mov   ax,VideoBufferSeg
      mov   es,ax
; ES:BX is BYTE address of pixel
      pop   ds
      ret

 PixelAddr256  ENDP

_TEXT    ENDS

_DATA    SEGMENT WORD PUBLIC 'DATA'

         EXTRN  PIXELS PER_LINE:word

_DATA    ENDS

         END
```

Figure 5 - Find Pixel Video Buffer Address

```
; File name:   SetPG256.asm
; Function: Set the value of a pixel
; in 256 colour mode
; Caller:   Microsoft C

;     void SetPixel256(x,y,n)
;     int x,y      pixel coordinates
;     int n        pixel colour
;
; stack frame addressing
ARGx     EQU   WORD PTR [bp+6]
ARGy     EQU   WORD PTR [bp+8]
ARGn     EQU   WORD PTR [bp+10]

_TEXT    SEGMENT  BYTE PUBLIC 'CODE'
         ASSUME   cs: TEXT,ds:_DATA

         EXTRN _PixelAddr256:FAR

      PUBLIC   _SetPixel256
_SetPixel256       PROC  FAR

      push  bp
      mov   bp,sp
      push  ds
      mov   ax,seg  DATA
      mov   ds,ax         ; DS -> DATA

      mov   ax,ARGy       ; AX := y
      cmp   ax,_CLIP_AREA_TOP
      jl exit
      cmp   ax,_CLIP_AREA_BOTTOM
      jg exit

      mov   bx,ARGx       ; BX := x
      cmp   bx, CLIP_AREA LEFT
      jl exit
      cmp   bx,_CLIP_AREA_RIGHT
      jg exit

      call  _PixelAddr256 ; ES:BX -> buffer
      mov   ax,ARGn
      and   ax,0ffh
      mov   cx,_WRITING_MODE
      jcxz  ReplacePixel
      dec   cx
      jcxz  AndPixel
      dec   cx
      jcxz  OrPixel
      dec   cx
      jcxz  XorPixel
      jmp   short ReplacePixel

AndPixel:
      and   es:[bx],al
      jmp   short exit
OrPixel:
      or es:[bx],al
      jmp   short exit
XorPixel:
      xor   es:[bx],al
      jmp   short exit
ReplacePixel:
      mov   es:[bx],al

exit:    pop   ds
      mov   sp,bp
      pop   bp
      ret

_SetPixel256   ENDP

_TEXT    ENDS

_DATA    SEGMENT WORD PUBLIC 'DATA'

         EXTRN _WRITING_MODE:WORD
         EXTRN _CLIP_AREA_LEFT:WORD
         EXTRN _CLIP_AREA_RIGHT:WORD
         EXTRN _CLIP_AREA_TOP:WORD
         EXTRN _CLIP_AREA_BOTTOM:WORD

_DATA    ENDS

         END
```

Figure 6 - SetPixel256

```
; File name:   SWMDG256.asm
; Function:    Set write mode.
; Caller:      Microsoft C

;     void SetWriteMode256(int mode)
;     int mode given as  0    replace
;                        1    AND
;                        2    OR
;                        3    XOR
;     held in WRITING_MODE
ARGmode  EQU   WORD PTR [bp+6]

_TEXT    SEGMENT  WORD PUBLIC 'CODE'
         ASSUME   cs:_TEXT,ds:_DATA

         PUBLIC    _SetWriteMode256
_SetWriteMode256  PROC  FAR

        push  bp
        mov   bp,sp
        push  ds
        mov   ax,seg  DATA
        mov   ds,ax    ; DS -> DATA

        mov   ax,ARGmode
        mov   _WRITING_MODE,ax

; return to caller

        pop   ds
        mov   sp,bp
        pop   bp
        ret

_SetWriteMode256  ENDP

_TEXT    ENDS

_DATA    SEGMENT WORD PUBLIC 'DATA'

         EXTRN  _WRITING_MODE:WORD

_DATA    ENDS

END
```

Figure 7 - SetWriteMode

Dividing this by 65536 (ie 64 KB) gives the bank number, the remainder being the offset within the bank. The internal routine `GetPixelAddress256`, Figure 5, calls `SetBank256` to do all the messy work, so does not itself have to take card types into account. This is used by `SetPixel256`, Figure 6, which has the added refinement of allowing pixels to be replaced, ANDed, ORed or XORed with those currently on the screen. The writing mode being set by `SetWriteMode256`, which is given in Figure 7.

Drawing Lines

We can now set a pixel anywhere on the screen. The next step is to draw lines.

At this point in my design, I had to strike a compromise between speed, flexibility and ease of coding. In the 256 colour modes, there is the additional complication of bank switching which, as I mentioned in the first article in this series, generally takes place at pixels part way along scan lines.

The easiest plan is to implement Bresenham's algorithm, using `SetPixel256()` for each pixel. This conveniently deals with switching banks at the correct place but is inefficient in calculating the address at every pixel. In his book *Programmer's Guide to PC and PS/2 Video Systems*, Richard Wilton presents a routine for non-bankswitching line-drawing which is much more efficient: it calculates a pixel address only at the start of the line, and uses offsets for subsequent pixels.

Horizontal lines are especially important in that they can be used for filling shapes such as rectangles, ellipses etc. For a horizontal line it is relatively easy to work out if bank switching is required and where to do it, since there can, at most, be one switching point. The technique is: first, calculate the address and bank at each end point; then, if the banks are the same, use a fast horizontal line technique as described by Wilton. If they differ, work out the address of the switching point and use the fast horizontal line algorithm twice, once for each

```
; File name:   HLinG256
; Function: Draw a horizontal line
; in 256 colour mode
; Caller:    Microsoft C

;      void HorizLine256(x1,x2,y,n)
;      int x1,x2  x coordinates of line
;      int y       y coordinate
;      int n       colour

;stack frame addressing
ARGx1     EQU     WORD PTR [bp+6]
ARGx2     EQU     WORD PTR [bp+8]
ARGy      EQU     WORD PTR [bp+10]
ARGn      EQU     BYTE PTR [bp+12]
x1off     EQU     WORD PTR [bp-2]
x1seg     EQU     WORD PTR [bp-4]
x2off     EQU     WORD PTR [bp-6]
x2seg     EQU     WORD PTR [bp-8]
VideoBufferSeg EQU    0a000h

_TEXT     SEGMENT  BYTE PUBLIC 'CODE'
          ASSUME   cs:_TEXT,ds:_DATA

          EXTRN _SetBank256:FAR

          PUBLIC    _HorizLine256
_HorizLine256  PROC  FAR

       push  bp
       mov   bp,sp
; get stack space for local variables
       sub   sp,8
       push  ds
       mov   ax,seg _DATA
       mov   ds,ax       ; DS -> DATA
       push  si
       push  di

; force x1 < x2

       mov   ax,ARGx1
       mov   bx,ARGx2
       cmp   ax,bx
       jle   L01
       xchg  ax,bx       ; AX <= BX

; clip to sides of window

L01:   cmp   ax,_CLIP_AREA_RIGHT
       jle   L02
       jmp   Exit
L02:   cmp   ax,_CLIP_AREA_LEFT
       jge   L03
       mov   ax,_CLIP_AREA_LEFT

L03:   cmp   bx,_CLIP_AREA_LEFT
       jge   L04
       jmp   Exit
L04:   cmp   bx,_CLIP_AREA_RIGHT
       jle   L05
       mov   bx,_CLIP_AREA_RIGHT
L05:   mov   ARGx1,ax
       mov   ARGx2,bx

; test clipping at top and bottom of window

       mov   ax,ARGy
       cmp   ax,_CLIP_AREA_TOP
       jl    Exit
       cmp   ax,_CLIP_AREA_BOTTOM
       jg    Exit

; clipping done

       mov   cx,ARGx2
       sub   cx,ARGx1   ; CX := x2 - x1

; now find if both end points
; lie in the same 64 KB segment

       push  cx          ; preserve CX
       mov   dx,_PIXELS_PER_LINE
       mul   dx
       add   ax,ARGx1
       jnc   L06
       inc   dx
L06:   mov   x1off,ax
       mov   x1seg,dx

       mov   ax,ARGy
       mov   dx,_PIXELS_PER_LINE
       mul   dx
       add   ax,ARGx2
       jnc   L07
       inc   dx
L07:   mov   x2off,ax
       mov   x2seg,dx

       sub   dx,x1seg
       jz OneSeg

; have to do the line in two sections
; corresponding to the two 64 KB
; segments it crosses, second bit first

       mov   dx,x2seg
       call   SetBank256
       mov   ax,VideoBufferSeg
       mov   es,ax
       mov   di,0     ; ES:DI -> buffer

       mov   cx,x2off
; CX is now the number of pixels-1
; right of the segment boundary
       inc   cx
       call  Set_Pixels

; now fix up cx for the first segment

       pop   cx
       sub   cx,x2off
       dec   cx
       push  cx

OneSeg:      mov   dx,x1seg
       call   SetBank256
       mov   ax,VideoBufferSeg
       mov   es,ax
       mov   di,x1off ; ES:DI -> buffer

       pop   cx
       inc   cx
; update the video buffer
       call  Set_Pixels

Exit:  pop   di
       pop   si
       pop   ds
       mov   sp,bp
       pop   bp
       ret

_HorizLine256  ENDP

Set_Pixels  PROC

; AL := pixel value
       mov   al,ARGn

       cmp   _WRITING_MODE,0
       je LREP
       cmp   _WRITING_MODE,1
       je LAND
       cmp   _WRITING_MODE,2
       je LOR
       cmp   _WRITING_MODE,3
       je LXOR

LREP:  ; update the video buffer
       rep   stosb
       ret

LAND:  and   es:[di],al
       inc   di
       loop  LAND
       ret

LOR:   or es:[di],al
       inc   di
       loop  LOR
       ret

LXOR:  xor   es:[di],al
       inc   di
       loop  LXOR
       ret

Set_Pixels  ENDP

_TEXT    ENDS

_DATA    SEGMENT WORD PUBLIC 'DATA'

         EXTRN  _CLIP_AREA_BOTTOM:WORD
         EXTRN  _CLIP_AREA_LEFT:WORD
         EXTRN  _CLIP_AREA_RIGHT:WORD
         EXTRN  _CLIP_AREA_TOP:WORD
         EXTRN  _PIXELS_PER_COLUMN:WORD
         EXTRN  _PIXELS_PER_LINE:WORD
         EXTRN  _WRITING_MODE:WORD

_DATA    ENDS

       END
```

Figure 8a - HorizLine256

```
; File name:   SLinG256.asm
; Function: Draw a short line
; contained in one 64 KB segment
; in 256 color mode
; Caller:   Microsoft C

; void ShortLine256(int x1,int y1,
                    int x2,int y2,int n)
; int x1,y1,x2,y2  pixel coordinates
; int n            pixel value

; stack frame addressing
ARGx1        EQU    WORD PTR [bp+6]
ARGy1        EQU    WORD PTR [bp+8]
ARGx2        EQU    WORD PTR [bp+10]
ARGy2        EQU    WORD PTR [bp+12]
ARGn         EQU    BYTE PTR [bp+14]
VARincr1     EQU    WORD PTR [bp-2]
VARincr2     EQU    WORD PTR [bp-4]
VARroutine   EQU    WORD PTR [bp-6]

 TEXT    SEGMENT  BYTE PUBLIC 'CODE'
      ASSUME   cs: TEXT,ds:_DATA

      EXTRN _PixelAddr256:FAR

      PUBLIC     ShortLine256
 ShortLine256  PROC   FAR

      push  bp
      mov   bp,sp
; get stack space for local variables
      sub   sp,6
      push  ds
      mov   ax,seg _DATA
      mov   ds,ax       ; DS -> DATA
      push  si
      push  di

; check for vertical line
      mov   si,_PIXELS_PER_LINE
      mov   cx,ARGx2
      sub   cx,ARGx1   ; CX := x2 - x1
      jz VertLine   ; jump if vertical

; force x1 < x2

      jns   L01     ; jump if x2 > x1
      neg   cx      ; CX := x1 - x2
      mov   bx,ARGx2   ; xchge x1 & x2
      xchg  bx,ARGx1
      mov   ARGx2,bx
      mov   bx,ARGy2   ; xchge y1 & y2
      xchg  bx,ARGy1
      mov   ARGy2,bx

; calculate dy = ABS(y2-y1)

L01:  mov   bx,ARGy2
      sub   bx,ARGy1 ; BX := y2 - y1
      jns   L03    ; jump if slope +ve
      neg   bx        ; BX := y1 - y2
      neg   si        ; negate y-increment

; select appropriate routine
; for slope of line

L03:  push  si ; preserve y-increment
    mov   VARroutine,offset LoSlopeLine
      cmp   bx,cx
      jle   L04  ; jump if dy <= dx
    mov   VARroutine,offset HiSlopeLine
      xchg  bx,cx  ; exchange dy and dx

; calculate initial decision variable
; and increments

L04:  shl   bx,1
      mov   VARincr1,bx
; incr1 := 2 * dy
      sub   bx,cx
      mov   si,bx
      sub   bx,cx
      mov   VARincr2,bx
; incr2 := 2 * (dy - dx)

; calculate first pixel address

      push  cx
      mov   ax,ARGy1 ; AX := y
      mov   bx,ARGx1 ; BX := x
      call  _PixelAddr256
      mov   di,bx    ; ES:DI -> buffer
      pop   cx
      inc   cx  ; = # pixels to draw
      pop   bx  ; BX := y-increment
; jump to appropriate routine for slope
      jmp   VARroutine


VertLine: ; routine for vertical lines

      mov   ax,ARGy1   ; AX := y1
      mov   bx,ARGy2   ; BX := y2
      mov   cx,bx
      sub   cx,ax      ; CX := dy
      jge   L31      ; jump if dy >= 0

      neg   cx       ; force dy >= 0
      mov   ax,bx    ; AX := y2

L31:
      inc   cx
      mov   bx,ARGx1   ; BX := x
      push  cx
      call  _PixelAddr256
      pop   cx
      mov   di,bx
; ES:DI -> video buffer
      mov   al,ARGn       ; AL := pixel value

      cmp   _WRITING_MODE,0
      je VREP
      cmp   _WRITING_MODE,1
      je VAND
      cmp   _WRITING_MODE,2
      je VOR
      cmp   _WRITING_MODE,3
      je VXOR

; update the video buffer
VREP: mov   es:[di],al
      add   di,si   ; goto next line
      loop  VREP
      jmp   Lexit

VAND: and   es:[di],al
      add   di,si
      loop  VAND
      jmp   short Lexit

VOR:  or es:[di],al
      add   di,si
      loop  VOR
      jmp   short Lexit

VXOR: xor   es:[di],al
      add   di,si
      loop  VXOR
      jmp   short Lexit


LoSlopeLine:
; routine for dy <= dx (slope <= 1)
; ES:DI -> video buffer
; BX = y-increment
; CX = #pixels to draw
; SI = decision variable

      mov   al,ARGn ; AL := pixel value

L11:  call  Set_Pixel   ; store pixel
      inc   di        ; increment x
      or si,si        ; test sign of d
      jns   L12     ; jump if d >= 0
      add   si,VARincr1
      loop  L11
      jmp   short Lexit

L12:  add   si,VARincr2
      add   di,bx    ; increment y
      loop  L11
      jmp   short Lexit


HiSlopeLine:
; routine for dy > dx (slope > 1)
; ES:DI -> video buffer
; BX = y-increment
; CX = #pixels to draw
; SI = decision variable

      mov   al,ARGn ; AL := pixel value

L21:  call  Set_Pixel
      inc   di            ; increment x
      add   di,bx         ; increment y

L22:  or si,si     ; test sign of d
      jns   L23       ; jump if d >= 0
      add   si,VARincr1
      dec   di
; decrement x (already incremented
; after call to Set_Pixel)
      loop  L21
      jmp   short Lexit

L23:  add   si,VARincr2
      loop  L21

Lexit:
      pop   di
      pop   si
      pop   ds
      mov   sp,bp
      pop   bp
      ret

 ShortLine256      ENDP

Set Pixel     PROC

      cmp     _WRITING_MODE,0
      je      SREP
      cmp     _WRITING_MODE,1
      je      SAND
      cmp     _WRITING_MODE,2
      je      SOR
      cmp     _WRITING_MODE,3
      je      SXOR

SREP: mov     es:[di],al
      ret

SAND: and     es:[di],al
      ret

SOR:  or es:[di],al
      ret

SXOR: xor     es:[di],al
      ret

Set_Pixel     ENDP

_TEXT     ENDS

_DATA     SEGMENT WORD PUBLIC 'DATA'

          EXTRN _PIXELS_PER_LINE:WORD
          EXTRN _WRITING_MODE:WORD

_DATA     ENDS

          END
```

Figure 8b - ShortLine256

section. The same approach would be possible for oblique lines but is quite complicated except in modes such as 1024 x 768 where bank switching takes place at the end of lines.

The compromise adopted in Video256 is to provide routines written in assembler for arbitrarily long horizontal lines, `HorizLine256()` (Figure 8a) and for short oblique (including vertical) lines which do not cross bank boundaries, `ShortLine256()` (Figure 8b). The `DrawLine256()` routine, which handles all cases, is written in C. It determines the type of line to be drawn and calls the assembler routines if it can. If not, it resorts to the Bresenham algorithm using `SetPixel256()`. Refer to Figure 8c for details.

Palettes

One of the reasons - perhaps the main reason - for going to all this trouble is to gain access to the 256 colours out of the 256K palette which the Super-VGA cards offer. After the effort required to make a mark on the screen, you could be forgiven for thinking that this would be a problem too.

Fortunately, the original IBM VGA modes included the oddball 320 x 200 x 256 colour mode 13h. In consequence, straightforward ways of setting up and reading palettes were defined by IBM, and all manufacturers follow them in order to remain compatible. Figure 9a gives suitable code for setting a palette register. It is written in C since speed is not crucial.

The number of possible different 256 colour palettes which could be chosen out of a total of 256K colours is truly astronomical, so there is plenty of scope for everyone to have their own favourite one as a default. `SetDefaultPalette256()`, Figure 9b, sets up the default which I generally employ. It gives a useful selection of colours which bear a relationship to the colour number. The eye is less sensitive to blue than it is to green and to red, so the palette

```
short Clip(short *x0,short *y0,
           short *x1,short *y1);
short SignInt(short i);
void ShortLine256(short x0,short y0,
                  short x1,short y1,
                  short colour);
void HorizLine256(short x0,short x1,
                  short y1,
                  short colour);
void SetPixel256(short x, short y,
                 short colour);
short  PixelsPerLine256(void);
void Swap(short *a, short *b);

extern short CLIP_AREA_LEFT;
extern short CLIP_AREA_RIGHT;
extern short CLIP_AREA_TOP;
extern short CLIP_AREA_BOTTOM;

void DrawLine256(short x0,short y0,
                 short x1,short y1,
                 short colour)
{ short i,x,y,dx,dy,xx0,yy0,xx1,yy1,e,
        s0,s1,temp,interchange,
        seg0,seg1;
  long pixels;

/* only clip when calling ShortLine256
   since HorizLine256 and SstPixel256
   clip to the window in the
   assemble code */

  dy=abs(y1-y0);
  if (dy==0)    /* a horizontal line */
  { HorizLine256(x0,x1,y0,colour);
    return;
  }

  x=x0; y=y0;
  pixels=(long)PixelsPerLine256();
  seg0=(pixels*y0+x0)>>16;
  seg1=(pixels*y1+x1)>>16;

  if (seg0==seg1)
/* line in one 64k segment so clip
   then use faster pure assembler
   routine */
  { xx0=x0; yy0=y0; xx1=x1; yy1=y1;
    if (Clip(&xx0,&yy0,&xx1,&yy1)!=0)
    { ShortLine256(xx0,yy0,
                   xx1,yy1,colour);
    }
    return;
  }
/* line in more than one 64k segment */
  dx=abs(x1-x0);
  s0=SignInt(x1-x0); s1=SignInt(y1-y0);
  if (dy>dx)
  { temp=dx; dx=dy; dy=temp;
    interchange=1;
  }
  else
    interchange=0;
  e=2*dy-dx;
  for (i=0;i<=dx;i++)
  { SetPixel256(x,y,colour);
    while (e>=0)
    { if (interchange==1)
        x=x+s0;
      else
        y=y+s1;
      e=e-2*dx;
    }
    if (interchange==1)
      y=y+s1;
    else
      x=x+s0;
    e=e+2*dy;
  }
}

short SignInt(short i)
{ if (i<0) return (-1);
  if (i>0) return (1);
  return (0);
}

short Clip(short *xx0,short *yy0,
           short *xx1,short *yy1)
{ short x0,y0,x1,y1;

  x0=*xx0; y0=*yy0; x1=*xx1; y1=*yy1;
  if (x0>x1)
  { Swap(&x0,&x1); Swap(&y0,&y1);
  }
  if ((x1<CLIP_AREA_LEFT)||
      (CLIP_AREA_RIGHT<x0)) return (0);
  if (x0!=x1)
  { if (CLIP_AREA_RIGHT<x1)
    { y1=(long)(CLIP_AREA_RIGHT-x0)*
                (y1-y0)/(x1-x0)+y0;
      x1=CLIP_AREA_RIGHT;
    }
    if (x0<CLIP_AREA_LEFT)
    { y0=(long)(CLIP_AREA_LEFT-x1)*
                (y0-y1)/(x0-x1)+y1;
      x0=CLIP_AREA_LEFT;
    }
  }

  if (y0>y1)
  { Swap(&x0,&x1); Swap(&y0,&y1);
  }
  if ((y1<CLIP_AREA_TOP)||
    (CLIP_AREA_BOTTOM<y0)) return (0);
  if (y0!=y1)
  { if (y0<CLIP_AREA_TOP)
    { x0=(long)(CLIP_AREA_TOP-y1)*
                (x0-x1)/(y0-y1)+x1;
      y0=CLIP_AREA_TOP;
    }
    if (CLIP_AREA_BOTTOM<y1)
    { x1=(long)(CLIP_AREA_BOTTOM-y0)*
                (x1-x0)/(y1-y0)+x0;
      y1=CLIP_AREA_BOTTOM;
    }
  }
  *xx0=x0; *yy0=y0; *xx1=x1; *yy1=y1;
  return (1);
}

void Swap(short *a, short *b)
{ short temp;

  temp=*a; *a=*b; *b=temp;
}
```

Figure 8c - DrawLine 256

allocates two bits to blue and three each to green and red.

The simplest scheme would be to allocate actual RGB levels linearly, 0 21 42 63 for blue and 0 9 18 27 36 45 54 63 for green and red. If you do this, you find that, in practice, the distribution of colours is not ideal. If you turn up the display brightness to improve the contrast between the dark colours, the light ones are all bleached out. Adjusting for the light colours results in the dark ones merging into near black. The reason for this is the high contrast of monitor displays, they are said to have a high *gamma*. Gamma correction, the compensation for this effect, is an important activity in any system dealing with a wide range of colours. `SetDefaultPalette256()` therefore, has a gamma correction parameter. The theoretical gamma for monitors working in a dark room is around 2.2. In a normal office environment the high ambient lighting level effectively reduces the contrast. From experience, a gamma of 1.6 is a reasonable compromise. Some monitors have significantly different gammas for red, green and blue. Usually the blue has the lowest contrast, so that an adjustment for a good high intensity white results in greys having a bluish cast. Finally note that if you wish to reset the whole of the palette it is much faster to use one BIOS call, as in `SetDefaultPalette256()` rather than setting registers individually.

```
#include <dos.h>
#include <math.h>

void SetPaletteReg256(short c, short r,
                      short g, short b)
{ union REGS reg;

  reg.h.ah=0x10;
  reg.h.al=0x10;
  reg.x.bx=c;
  reg.h.ch=g; reg.h.cl=b; reg.h.dh=r;
  int86(0x10,&reg,&reg);
}
```

Figure 9a - SetPaletteReg256

Fundamental to all drawing operations is the calculation of pixel addresses

Graphics Text

Few graphics programs entirely avoid having to display text. Many will mix text with graphics output, and at the very least they issue prompts and receive keyboard input.

In the alphanumeric video modes text is easy. The required character and its attributes are loaded into the video buffer and the hardware character generator does the rest, placing the pixels onto the screen. In graphics modes, however, it is your program which has to do the work of writing every pixel of every character into the video buffer. C programmers will already have encountered this problem. The customary `printf()` function used in text modes must be replaced by the much more clumsy `_outtext()` when using graphics. Of course, doing it yourself does give you much greater flexibility in return for the greater effort. In particular, graphics modes characters can be positioned to the nearest pixel location, not just to a coarse character grid such as 80 x 25. The colour and writing mode of individual pixels is also under your control, allowing for special effects.

```
#include <dos.h>
#include <math.h>

void SetDefaultPalette256(float gamma)
{ union REGS reg;
  struct SREGS sreg;
  short rr,gg,bb,r,g,b,i;
  unsigned char *PalettePtr,
                Palette[256][3];
  double gam;

  gam=1.0/gamma; i=0;
  for (b=0;b<4;b++)
  { bb=63.0*pow((double)(b/3.0),gam);
    for (g=0;g<8;g++)
    { gg=63.0*
            pow((double)(g/7.0),gam);
      for (r=0;r<8;r++)
      { rr=63.0*
            pow((double)(r/7.0),gam);
        Palette[i][0]=rr;
        Palette[i][1]=gg;
        Palette[i][2]=bb;
        i++;
      }
    }
  }
  reg.h.ah=0x10;
  reg.h.al=0x12;
  reg.x.bx=0;
  reg.x.cx=256;
  PalettePtr=(char *)Palette;
  sreg.es=FP_SEG(PalettePtr);
  reg.x.dx=FP_OFF(PalettePtr);
  int86x(0x10,&reg,&reg,&sreg);
}
```

Figure 9b - SetDefaultPalette256

```
; File name:   Charg256.ASM
; Function: Draw a character
; in 256-color mode
; Caller:   Microsoft C
;
;void DrawChar256(c,x,y,fgd,bkgd)
;     int c          character code
;     int x,y        upper left pixel
;     int fgd,bkgd   foreground and
;                    background pixel
;                    values
; if background = -1
;              don't set the pixel

; .286 used only for pusha and popa
      .286

; stack frame addressing
ARGc        EQU    WORD PTR [bp+6]
ARGx        EQU    WORD PTR [bp+8]
ARGy        EQU    WORD PTR [bp+10]
ARGfgd      EQU    BYTE PTR [bp+12]
ARGbkgd     EQU    WORD PTR [bp+14]
PixLine8    EQU    WORD PTR [bp-2]
Posx        EQU    WORD PTR [bp-4]

_TEXT    SEGMENT  BYTE PUBLIC 'CODE'
         ASSUME   cs:_TEXT

         EXTRN _PixelAddr256:FAR
         EXTRN _SetPixel256:FAR

         PUBLIC     _DrawChar256
_DrawChar256    PROC   FAR

      push  bp
      mov   bp,sp
      sub   sp,4
      push  si
      push  di
      push  ds

      mov   bx,ARGx     ; BX := x

; need a copy of the x-start position
; for each line
      mov    Posx,bx

; set up character definition
; table addressing

      mov    ax,40h ; = BIOS Video seg
      mov    ds,ax  ; display data area
      mov    cx,ds:[85h]
; CX = POINTS (pixel rows in character)

      xor    ax,ax
      mov    ds,ax  ; = absolute zero

      mov    ax,ARGc ; AL := char code
      mov    bx,43h*4
; DS:BX -> int 43h vector if char < 80h
      lds    si,ds:[bx]
; DS:SI -> start of character table
      mul    cl
; AX := offset into char def table
; (POINTS * char code)
      add    si,ax
; SI := addr of char def

; store the char in the video buffer

      mov    bx,ARGbkgd
      mov    bh,bl
; BH := background pixel value
      mov    bl,ARGfgd
; BL := foreground pixel value

L10:  push   cx ; keep CX across loop
      mov    cx,8  ; = char width
      lodsb
      mov    ah,al
; AH := bit pattern for next pixel row

L11: mov    al,bl
; AL := foreground pixel value
      shl    ah,1  ; shift into CF
      jc L12
; jump if bit pattern specifies a
; foreground pixel (bit = 1)

      cmp    ARGbkgd,-1  ; skip out?
      je L13       ; 'colour' is -1

      mov    al,bh
; AL := background pixel value
L12:     pusha          ; to be safe!
      push  ax         ; pixel value
      mov   ax,ARGy    ; y coord
      push  ax
      mov   ax,Posx
      push  ax         ; x coord
      call  _SetPixel256
      pop   ax
      pop   ax
      pop   ax
      popa
L13:  inc   Posx       ;bump xcoord
      loop  L11

      mov   ax,ARGy
      inc   ax
      mov   ARGy,ax    ;increment y
      mov   ax,ARGx    ; re-set x coord
      mov   Posx,ax
      pop   cx
      loop  L10   ; loop down character

      pop   ds
      pop   di
      pop   si
      mov   sp,bp
      pop   bp
      ret

_DrawChar256    ENDP

_TEXT     ENDS

      END
```

Figure 10 -DrawChar256

```
extern short CLIP_AREA_LEFT;
extern short CLIP_AREA_RIGHT;
extern short CLIP_AREA_TOP;
extern short CLIP_AREA_BOTTOM;

short DrawString256(unsigned char *stg,
                    short x, short y,
                short frgd, short bkgd)

/* Draw as much as possible of the
   character string stg and return
   the number of characters drawn
   in whole or part */

{ short i,nchar,xloc;

  nchar=0;
  if ((CLIP_AREA_TOP>=(y+15)) &&
                 (y>=CLIP_AREA_BOTTOM))
  { xloc=x;
    for (i=0;i<strlen(stg);i++)
    { if ((xloc<=CLIP_AREA_RIGHT) &&
          (CLIP_AREA_LEFT<=(xloc+7)))
      { DrawChar256((int)(*(stg+i)),
                xloc,y,frgd,bkgd);
        xloc+=8; nchar++;
      }
    }
  }
  return(nchar);
}
```

Figure 11 - DrawString256

```
short Input256(unsigned char *prompt,
               short x, short y,
               short frgd, short bkgd,
               unsigned char *response)
{ unsigned char ch;
  short i=0,loc;

  DrawString256(prompt,x,y,frgd,bkgd);
  loc=x+(strlen(prompt)<<3);
  do
  { ch=getch();
    if ((ch==8)&&(i>0))
    { loc-=8; i--;
      DrawChar256(32,loc,y,frgd,bkgd);
    }
    else if (ch>=32)
    { *(response+i)=ch;
      DrawChar256(ch,loc,y,frgd,bkgd);
      i++; loc+=8;
    }
  }
  while (ch!=13);
  *(response+i)='\0';
  return (i);
}
```

Figure 12 - Input256

The most convenient way of handling the displayable character set is to hold them in a table of bit patterns representing the character's pixel patterns. Usually, these patterns are all made to fit into the same size rectangular matrix and have one bit per pixel. This is not essential, the characters and patterns could be of variable size - but this complicates the software character generator.

Because of this great flexibility, you may like to experiment with your own characters and escape from the staid design provided by IBM. For the purposes of *this* exercise, however, I have kept it simple, and made use of the in-built character definitions.

The default VGA characters are defined on an eight wide by 16 tall matrix. The table contains definitions for all 256 characters. It is directly accessible, its address obtained by calling Int 10h, function 11h, with AL=30h. Moreover, when a graphics mode is initialised, the address of the table is copied into interrupt vector 43h. The code of Figure 10 uses these ideas. It is closely modelled on examples in Wilton's book, with some modifications and additions to allow for bank switching.

Figure 11 is a C function for outputting character strings. It takes the simplest possible approach to clipping characters at the screen boundary. `DrawChar256()` is called for any character at least partially within the clipping area. It then relies on the routine `SetPixel256()` in `DrawChar256()` to do the fine clipping.

Finally, a simple routine for keyboard input in graphics mode. Remember that functions such as C's `scanf()` will not work, because of the echoing of characters to the screen. The routine `Input256()`, which uses `getch()` and `DrawChar256()` gets round this problem.

EXE

Dr Graeme Webster was formerly Head of Department of Computer Science and later Deputy Director, Academic, of Teesside Polytechnic. He has been involved with computer graphics for the last 20 years with an especial interest in 3D visualisation for Designers. He is currently setting up a Centre for Scientific Visualisation under the aegis of the Teesside Development Corporation.

To obtain a copy of the source code, please send a blank, formatted disk and a stamped, self-addressed envelope as described in the 'Editorial' notes on Page 1. Follow the instructions on Page 1 exactly, or we may not be able to return your disk. Mark envelopes 'SUPER VGA'.

If you are interested in learning more about the less exotic VGA modes, Dr Webster recommends Programmer's Guide to PC and PS/2 Video Systems (by Richard Wilton, pub Microsoft Press, £22.95. ISBN 1-55615-103-9).

A Studied Spontaneity

Until the object-oriented assembler arrives, assembly language programmers will have to stick to function libraries. Robert Schifreen looks at one of the biggest.

The problem with programming is that you seem to spend half your life re-inventing wheels. Although you can reuse code from previous programs, it's never easy to take a whole subroutine and incorporate it into a new project without a large degree of tinkering. And if you tinker for too long, you end up with a much-improved version of the routine which you then feel compelled to include in the original program, and so the development cycle goes on and on.

To help overcome this problem, C programmers have the concept of object-oriented programming *(Shurely 'C++ programmers' - Ed)*. Personally, though, I tend not to use C for much of my programming. I consider C as a language that's neither high-level nor low-level. Therefore, for low-level work (which accounts for 90% of my programming) I prefer to stick to a language that was designed for the job, namely assembly language. For true high-level jobs, I go for compiled Microsoft BASIC or 4GLs such as dBASE or even a spreadsheet.

For assembly language programmers, the cure for WRS (Wheel Re-invention Syndrome) is a function library. If it's a general library you want, rather than a specialist one that concentrates solely on graphics or comms, then Spontaneous Assembly is well worth looking at.

What is it?

Spontaneous Assembly, from Base Two Development, is a library of almost 700 routines written in, and designed for use by, assembly language programmers using MS-DOS. The package includes the full source code, all of which is in assembly language too, so you can customise the package if you want to. This is not advisable, however - the whole point about having a collection of ready-made functions at your fingertips is that you can use them straight away. The manual says that you can copy chunks of the source code into your own programs, but I'd suggest that it's easier to LINK in the required routines when you want them, unless you want to make major alterations.

As far as copyright goes, you are free to include any of the SA routines in your own code, as long as they are distributed only in executable form. You are also obliged to put a copyright notice in the code and manuals. There's no need to put Acclaim Technologies' (Base Two is a division of Acclaim) copyright notice in there; your own will suffice. One minor point. You aren't permitted to use SA to create a package which 'substantially duplicates the overall functionality' of SA. This is a common clause nowadays - even the latest Borland licence states that you can't use Borland C to write a commercial C compiler.

Getting Started

The package consists of two or four disks, depending on the size of your drive. Both sizes of disk are supplied as standard, along with a manual in the style of a paperback book. If you think that the manual is suitable material for bedtime reading, make sure that your springs can stand the weight - the thing runs to 750 pages and is two inches thick.

Figure 1 - Inteser Maths Functions

ABS_D	- Converts a signed dword to its absolute value.
ABS_Q	- Converts a signed quadword to its absolute value.
NEG_Q	- Negates a signed quadword.
ADD_QD	- Adds an unsigned dword into an unsigned quadword.
ADD_QQ	- Adds two unsigned quadwords.
ADDS_QQ	- Adds two signed quadwords.
ADD_QW	- Adds an unsigned word into an unsigned quadword.
ADDS_DW	- Adds a signed word into a signed dword.
ADDS_QD	- Adds a signed dword into a signed quadword
ADDS_QW	- Adds a signed word into a signed quadword.
CMP_DD	- Compares two unsigned dwords.
CMPS_DD	- Compares two signed dwords.
CMP_QD	- Compares an unsigned quadword against an unsigned dword.
CMP_QQ	- Compares two unsigned quadwords.
CMPS_QQ	- Compares two signed quadwords.
CMP_QW	- Compares an unsigned quadword against an unsigned word.
CMPS_DW	- Compares a signed dword against a signed word.
CMPS_QD	- Compares a signed quadword against a signed dword.
CMPS_QW	- Compares a signed quadword against a signed word.
DEC_Q	- Decrements a quadword.
DIV_DW	- Divides an unsigned dword by an unsigned word, returning an-unsigned dword result.
DIV_QD	- Divides an unsigned quadword by an unsigned dword, returning an-unsigned quadword result.
DIV_QQ	- Divides an unsigned quadword by an unsigned quadword, returning an-unsigned quadword result.
DIV_QW	- Divides an unsigned quadword by an unsigned word, returning an-unsigned quadword result.
DIVR_DW	- Divides an unsigned dword by an unsigned word, returning a rounded-unsigned dword result.
DIVR_QD	- Divides an unsigned quadword by an unsigned dword, returning a-rounded unsigned quadword result.
DIVR_QQ	- Divides one unsigned quadword by another, returning a - rounded unsigned quadword result.
DIVR_QW	- Divides an unsigned quadword by an unsigned word, returning a rounded-unsigned quadword result.
DIVRS_DW	- Divides a signed dword by a signed word, returning a rounded-signed dword result.
DIVRS_QD	- Divides a signed quadword by a signed dword, returning a - rounded signed quadword result.
DIVRS_QQ	- Divides one signed quadword by another, returning a rounded-signed quadword result.
DIVRS_QW	- Divides a signed quadword by a signed word, returning a rounded-signed quadword result.
DIVRS_WB	- Divides a signed word by a signed byte, returning a rounded-signed word result.
DIVR_WB	- Divides an unsigned word by an unsigned byte, returning a rounded-unsigned word result.
DIVS_DW	- Divides a signed dword by a signed word, returning a - signed dword result.
DIVS_QD	- Divides a signed quadword by a signed dword, returning a - signed quadword result.
DIVS_QQ	- Divides one signed quadword by another, returning a - signed quadword result.
DIVS_QW	- Divides a signed quadword by a signed word, returning a - signed quadword result.
DIVS_WB	- Divides a signed word by a signed byte, returning a - signed word result.
DIV_WB	- Divides an unsigned word by an unsigned byte, returning an - unsigned word result.
INC_Q	- Increments a quadword.
MOV_QD	- Moves the value of an unsigned dword into a quadword.
MOV_QQ	- Moves the value of an unsigned quadword into another quadword.
MOVS_QQ	- Moves the value of a signed quadword into another quadword.
MOV_QQES	- Moves the value of an unsigned quadword into another quadword-in another segment.
MOVS_QQES	- Moves the value of a signed quadword into another quadword - in another segment.
MOV_QW	- Moves the value of an unsigned word into a quadword.
MOVS_QD	- Moves the value of a signed dword into a quadword
MOVS_QW	- Moves the value of a signed word into a quadword.
MUL_DW	- Multiplies an unsigned dword by an unsigned word, returning an-unsigned dword result.
MUL_QD	- Multiplies an unsigned quadword by an unsigned dword, returning an-unsigned quadword result.
MUL_QQ	- Multiplies one unsigned quadword by another, returning an - unsigned quadword result.
MULS_QQ	- Multiplies one signed quadword by another, returning a signed-quadword result.
MUL_QW	- Multiplies an unsigned quadword by an unsigned word, returning an-unsigned quadword result.
MULS_DW	- Multiplies a signed dword by a signed word, returning a - signed dword result.
MULS_QD	- Multiplies a signed quadword by a signed dword, returning a-signed quadword result.
MULS_QW	- Multiplies a signed quadword by a signed word, returning a - signed quadword result.
RAND_INIT	- Initialises the random number generator.
RANDR_W	- Generates a random number within a specific range.
RAND_W	- Generates a random number.
SUB_QD	- Subtracts an unsigned dword from an unsigned quadword.
SUB_QQ	- Subtracts one unsigned quadword from another.
SUBS_QQ	- Subtracts one signed quadword from another
SUB_QW	- Subtracts an unsigned word from an unsigned quadword.
SUBS_DW	- Subtracts a signed word from a signed dword.
SUBS_QD	- Subtracts a signed dword from a signed quadword.
SUBS_QW	- Subtracts a signed word from a signed quadword.

The routines themselves are divided into a number of categories. In alphabetical order, these are: array management, buffer manipulation, character classification and conversion, console I/O, data conversion, date and time manipulation, DOS console I/O, file and directory management, file I/O, integer maths, memory management, memory manipulation, miscellaneous DOS, miscellaneous hardware, program and environment control, string manipulation, table management and windowing. Needless to say, there's no room here to cover all of these, so I'll stick to the ones that potentially will save you the most time.

Installation of SA took about 10 minutes. It involved making half a dozen directories on my hard disk, and copying a number of .EXE files from the floppies. There is no copy protection. Running the .EXE files produced the SA package itself. The .EXE files are self-unpacking files which are compressed in order to save disk space. A previous version of SA, which I've had for around six months, took nine 5.25 inch disks rather than four. I don't object to companies saving money this way, but an INSTALL program would have been appreciated. Even so, I installed the thing fairly quickly and was soon ready to go. I decided not to install the source code for the routines, and I opted to stick with the tiny memory model for my experiments (tiny, small, medium, compact and large are available). With this configuration, the system took 680 KB of hard disk space, including a few example files.

Getting Productive

There are a number of sample files supplied with the package. I decided to try WIN.ASM, which claimed to demonstrate the windowing functions. I did a SET LIB to point to SA's LIB directory, and a SET INCLUDE to point to the include files. With these in place, creating WIN.COM from WIN.ASM was simply a matter of typing `MASM /d_model=tiny win;` followed by `LINK startt+win,win,,sat;`. The `/d` switch in the first statement tells MASM to define the value of `_model` as tiny, while `startt` and `sat` in the second statement point the linker towards the tiny versions of the startup code and the SA library. After running EXE2BIN, I had a .COM program which, when run, put up some text in a coloured window in the middle of the screen. I could move this around the screen with the cursor keys, and it even did those posh grey shadows like the Norton Utilities. All this in just 7916 bytes of .COM file, and only 20 minutes after I started installing the package.

Figure 2 - String routines available

STR_CAT	- Appends one string to the end of another.
STR_CATE	- Appends one string to the end of another, returning the address of the end of the modified string.
STR_CHR	- Searches a string for the first occurrence of a character.
STR_CHRI	- Searches a string for the first occurrence of a character, ignoring case.
STR_CHRN	- Searches a string for the first occurrence of a character which is NOT the given character.
STR_CHRNI	- Searches a string for the first occurrence of a character which is NOT the given character, ignoring case.
STR_CMP	- Compares one string against another.
STR_CMPI	- Compares one string against another, ignoring case.
STR_CPY	- Copies a string to another location.
STR_CPYE	- Copies a string to another location, returning a pointer to the end-of the copy.
STR_CSPN	- Determines the size of the front of a string where none of the-characters match any of those in a set of characters.
STR_DRVPATH	- Locates a drive specifier and path substring in a pathname string.
STR_DRIVE	- Locates a drive specifier substring in a pathname string.
STR_END	- Returns the address of the end of a string.
STR_ENDDI	- Returns the address of the end of a destination string.
STR_EXT	- Locates a filename extension substring in a string.
STR_FNAME	- Locates a filename substring in a string.
STR_FNAMEXT	- Locates a filename/extension substring in a string.
STR_LEN	- Returns the length of a string.
STR_LENDI	- Returns the length of a destination string.
STR_LWR	- Converts upper-case characters in a string to lower-case.
STR_NCAT	- Appends a portion of one string to the end of another.
STR_NCATE	- Appends a portion of one string to the end of another, returning-the address of the end of the modified string.
STR_NCMP	- Compares a portion of one string to a portion of another.
STR_NCMPI	- Compares a portion of one string to a portion of another, ignoring-case.
STR_NCPY	- Copies a portion of a string to another location.
STR_NSET	- Sets the first "n" characters of a string to a specified character.
STR_PATH	- Locates a path substring in a string.
STR_PBRK	- Scans a string for the first occurrence of any character from a-set of characters.
STR_PBRKN	- Scans a string for the first occurrence of any character which is NOT-in a set of characters.
STR_RCHR	- Searches a string for the last occurrence of a character.
STR_RCHRI	- Searches a string for the last occurrence of a character, - ignoring case.
STR_RCHRN	- Searches a string for the last occurrence of a character which is-NOT the given character.
STR_RCHRNI	- Searches a string for the last occurrence of a character which is-NOT the given character, ignoring case.
STR_REV	- Reverses the order of the characters in a string.
STR_SET	- Sets all characters in a string to a specified character.
STR_SKIPS	- Skips past space characters in a string.
STR_SKIPW	- Skips past white-space characters in a string.
STR_SPN	- Determines the size of the front of a string where all of the-characters match one of those in a set of characters.
STR_STR	- Searches a string for the first occurrence of a substring.
STR_STRI	- Searches a string for the first occurrence of a substring, ignoring-case.
STR_TOKS	- Starts a tokenising process on a string, which is carried out-by STR_TOK.
STR_TOK	- Breaks a string into tokens by matching characters in the string-with a set of delimiting characters.
STR_TRIMS	- Removes space characters from the end of a string.
STR_TRIMW	- Removes white-space characters from the end of a string.
STR_UPR	- Converts lower-case letters in a string to upper-case.

A word on assemblers and linkers. I used Microsoft MASM 5.1 and Microsoft LINK 5.05. TASM, from Borland, works too. Previous versions of MASM are said to have problems. I'm normally a user of OPTASM from SLR Systems, as it's always proved 100% MASM compatible yet around four times faster. Unfortunately, SA seemed to confuse OPTASM, and the simple windowing example mentioned above produced about 50 error messages at assembly time. Using OPTASM's full-MASM-compatibility switch, I managed to reduce the number of errors by half, but couldn't get anything to assemble properly.

The Skeleton

Assuming (which we will, for a moment) that you're writing purely in assembler, every program starts off in the same way. You simply copy MAIN.XXX to the name of your choice (changing the extension to .ASM), and then start adding your own code. The skeleton file contains all the directives and setup code required to handle calls to the SA library. If you want to keep a code module in a separate source file, then there's a skeleton file called MODULE.XXX which lets you do this.

Creating a Hello World program involves just a few amendments to the MAIN file. First, `put_str` needs to be added to the list of `extrn` definitions at the top of the file. All routines that you call from the SA library must be defined external in this way. Next, in the area marked for data, I added:

```
message db 'Hello World!'
    db 0Dh,0Ah,0
```

Note that `put_str` requires a zero-terminated string, rather than the standard DOS function call that expects a dollar sign at the end.

Finally, between `.PROC MAIN` and `.ENDP MAIN`, the code itself. This consists of just 2 lines:

```
mov si,offset @dataseg:message
call put_str
```

The file assembled first-time, and HELLO.COM was 75 bytes long.

Getting Adventurous

Getting slightly more daring, I decided to try a call to SA's version of the EXEC call. My current project under development needs to be able to load and execute a program. Using DOS to do this is fairly complex, as you have to ensure that there is enough memory to hold the incoming code, and that everything is set up correctly and then restored afterwards. Spontaneous Assembly made it all easy. All I had to do was to put the program's name in a string pointed to by es:di, and any command line argu-

Figure 3 - Text window routine available

DRAW_WBORDER	-	Draws a border within the active window.
WBORDER_ON	-	Enables border access within the active window.
WBORDER_OFF	-	Disables border access within the active window.
WIN_BUFSIZE	-	Returns the size of the buffer required by a window.
WIN_CLOSE	-	Closes a window.
WIN_CREATE	-	Creates a window.
GET_BACKDROP	-	Returns the windowing backdrop character and attribute.
GET_BACKDROPBUF	-	Returns the address of the current windowing backdrop buffer.
GET_WHANDLE	-	Returns the handle of a window at a specified level.
GET_WLEVEL	-	Returns the level of a window.
GET_WPOS	-	Returns the position of a window.
GET_WSPEC	-	Returns the specifications of a window.
WIN_INIT	-	Initialises the windowing system.
WIN_MOVE	-	Moves a window.
WIN_OPEN	-	Opens a window.
WIN_REMOVE	-	Removes a window.
SET_BACKDROP	-	Sets the windowing backdrop character and attribute.
SET_BACKDROPBUF	-	Installs a windowing backdrop buffer.
SET_WBORDER	-	Defines the border attribute and characters for a window.
SET_WLEVEL	-	Sets the level of a window.
SET_SPOS	-	Sets a window shadow position.
SET_WPOS	-	Sets the position of a window.
WIN_SELECT	-	Selects the active window.
SET_WSHADOW	-	Sets the shadow characters and attributes of a window.
WIN_SUSPEND	-	Temporarily suspends windowing.
WIN_RESUME	-	Resumes windowing.
UPDATE_CURSOR	-	Updates the hardware cursor position within the windowing system.
UPDATE_RECT	-	Updates a rectangular region of the screen.
UPDATE_SCREEN	-	Updates the entire screen.
UPDATE_WINDOW	-	Updates a window.
UPDATE_OFF	-	Turns auto update off.
UPDATE_ON	-	Turns auto update on.

ments in ds:si. Then, a single call to `EXEC_PROG` loaded and executed the file. Any .COM or .EXE file can be executed this way and the example I chose (WordPerfect 5.1) worked first time. If the program whose name is supplied in es:di is not found in the current directory, SA sets the carry flag to indicate an error. If you want SA to search the DOS path, and also to handle batch files as well, `EXEC_SYS` is the call to use. However, this works by loading a second copy of COMMAND.COM while `EXEC_PROG` does not.

There are no fewer than 10 copies of the SA library on the supplied disks. There's one for each of the five supported memory models, as already mentioned, plus another version of each file linked in such a way as to be callable from Turbo and Microsoft C. Even so, you can't `LINK` these routines with your C code, but you have to resort to inline assembly language to set up the segments and make the call. This is far from neat, but it's very efficient and adequately covered in the manual.

Conclusion

Spontaneous Assembly is an excellent package, and a useful addition to any DOS programmer's toolbox. Throughout the two weeks that I used it, it appeared solid and stable. As a general purpose library, it's superb, though it lacks support in specialist areas like graphics (the windows are all text-based), communications and complex numbers. For these facilities, you need to look for a special purpose package, of which there are many. Printer support is also non-existent.

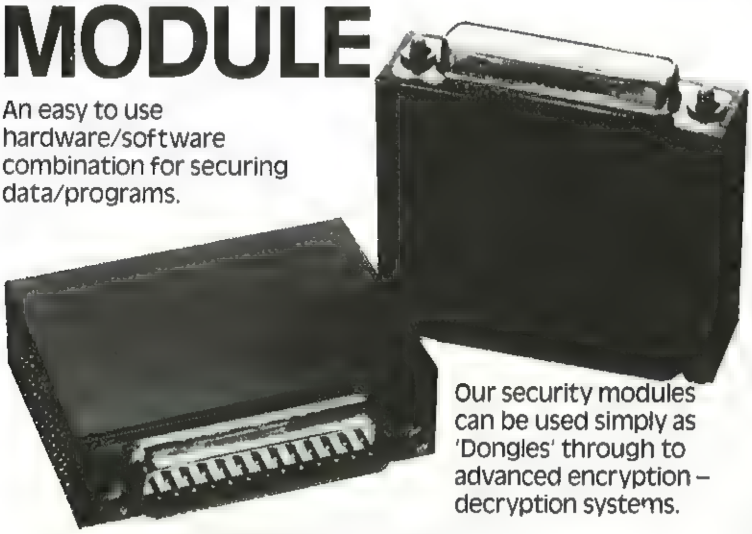
The SA documentation is excellent; there is a clear description of each routine, what it does, and how to call it. Some online documentation in the form of a Norton Guide would have been appreciated. I assume that its omission is linked to the size of the manual, in that both are forms of copy protection.

Unfortunately it's not possible to list all the functions available in a review like this. However, Figures 1, 2 and 3 list those routines connected with integer maths, string manipulation and text-based windows. The list includes only those routines which are documented in the SA manual. There is an equal number of undocumented routines which are designed to be called by the documented ones. For example, `GOTO_XY` calls `GOTO_X` and `GOTO_Y` separately. Although the undocumented routines are not in the manual, they are fully documented in the source code listing, and are identified by having names that start with a dollar sign. The listings in Figures 2 to 4, by the way, were compiled from the source code listings.

I'd recommend a copy of Spontaneous Assembly to any programmer who wants instant access to a megabyte of source code, in a form that can be modified or called directly. After all, what's the point of spending weeks writing a set of subroutines if someone has already done the work and is willing to sell it to you for less than the cost of a day's coding?

EXE

Robert Schifreen was Editor of .EXE between November 1987 and April 1990. He now runs a small publishing company called TTK Ltd by day, and slaves over a hot assembler by night. Spontaneous Assembler is available in the UK from the Software Construction Company (0763 244114) for £255, which compares with the US price of $395. Base Two Development are on 0101 801 222 9500, should you have any technical enquiries.



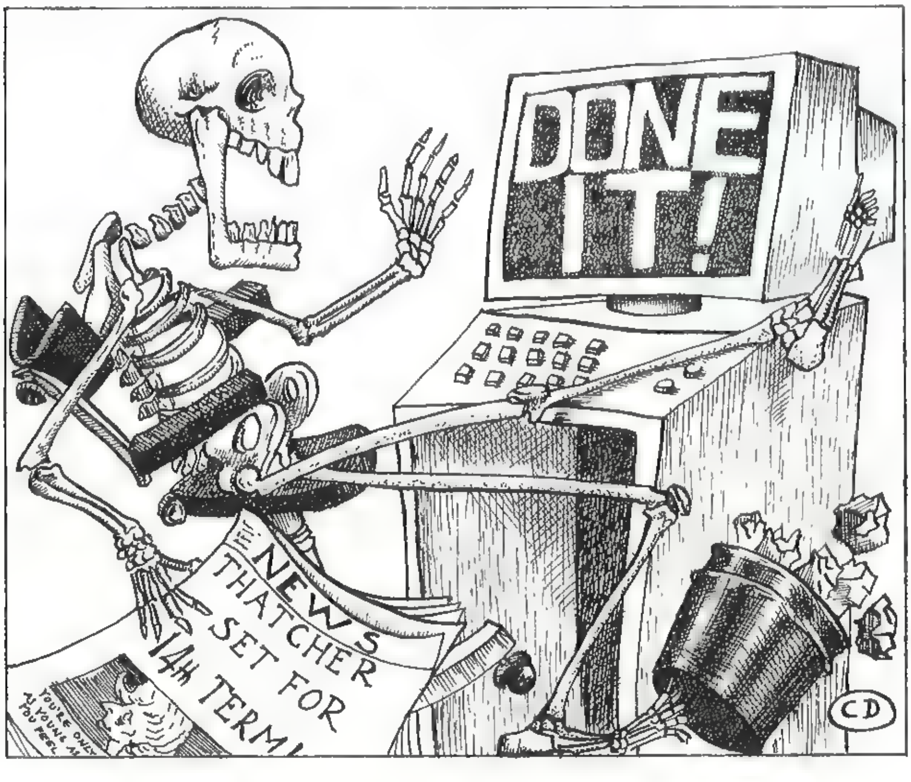
Not enough Time in the Universe

'Leave the machine churning overnight, it will probably be done by morning.' Bad advice if you are tackling an NP-complete problem, as Darrell Ince explains.

There is a tendency to think of computers as omnipotent: that no matter what problem we give them, they will be capable of a solution. However, in the last few years, we have seen this assumption completely overturned. Research scientists have discovered problems that are incapable of being efficiently solved: that no matter how big a computer you employ, even small instances of a problem will defeat it. Such problems are called NP-complete.

Before examining the nature of NP completeness, it is worth looking at a branch of computing known as algorithmic complexity. Researchers working in this field are trying to develop techniques which lead to the derivation of some algebraic description of the run-time or memory performance of a program. They have devised a notation, known as the 'O notation', which describes the run-time behaviour of a program.

For example, if the run-time of a program rises linearly with the amount of data that it processes then its algorithmic complexity is written as $O(n)$. This means that if a program runs in m seconds with a items of data, it will run in $2m$ seconds with $2a$ items of data. This is reasonably respectable behaviour. It is certainly better than a program which has a complexity of $O(n^2)$, where the increase in run-time varies as the square of the size of the problem.



An example of a very primitive way of determining algorithmic complexity is as follows. Consider the Pascal fragment shown below, which just initialises a two-dimensional array to zero:

```
for i:= 1 to n do
  for j:=1 to n do
    a[i,j] := 0;
```

The second `for` statement is executed n times, so clearly the assignment statement is executed n^2 times. This gives the algorithmic complexity of this simple array assignment as $O(n^2)$. This is a rather trivial example, but it does give the flavour of the sort of analysis that researchers carry out. For a meaningful program, however, the mathematics can be quite horrendous.

American researchers have been at the forefront in complexity research. They have provided a number of mathematical tools which enable a software developer to characterise the run-time of a program in algebraic terms. During the 1970s, one young researcher, Steven Cooke, made an extraordinary discovery. He found that there were a number of problems whose increase in run-time as a function of the amount of data processed was such that their growth was exponential. A small instance of a problem could be solved in a few seconds, the next bigger instance in an hour or two and the next instance in a few days even using a supercomputer. He showed that there were some problems which could not be solved with realistic examples within the known future life of the universe. The explosive growth in such problems is shown in Figure 1.

A problem such as this has a complexity of O(exp n). At first, in the early 1970s, there was the temptation to dismiss the work on NP-complete problems as something of an academic plaything. However, very rapidly, with the growth of computing power and the ambition of the computing industry to take on larger and larger challenges, a number of problems were discovered to be NP-complete. These problems came from diverse application areas including VLSI fabrication, operations research, molecular biology and network scheduling.

A typical NP-complete problem occurs in delivery systems. You are given a whole series of containers of varying sizes; the aim is to pack a lorry or van with these containers such that the minimum amount of space is empty.

Travelling Salesman

Probably the most famous NP-complete problem is the travelling salesman problem. Figure 2 shows a network of cities {a,b,c,d,e,f} with a number of roads joining them. Each road has an associated mileage, and the aim is to find a tour of *all* the cities, such that the mileage is minimised.

Small instances of the travelling salesman problem can be solved in seconds. Larger (more practical) instances, involving 50 cities and hundreds of routes, require years of computation. This is a tragedy, as variants of this problem occur in many areas, such as the design of fluid transmission systems, delivery scheduling and VLSI design.

One of the most interesting features of NP-complete problems is that they are based on what might seem a shaky edifice. They possess this feature: if you can write an efficient program which solves one of the large collections of currently known NP-complete problems, then all the problems are capable of an efficient solution. Certainly, if anybody managed to find an efficient program then it would be worth a Nobel prize.

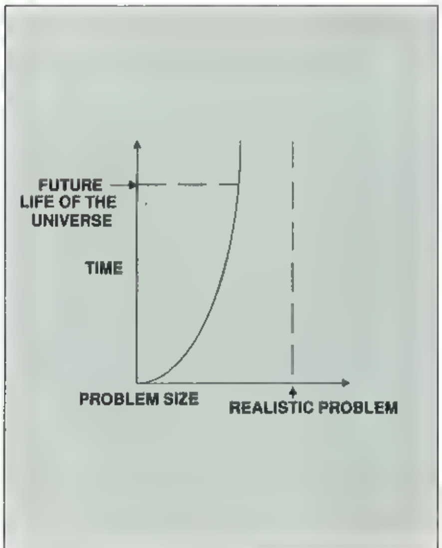


Figure 1 - The growth of an NP-complete problem.

A probablistic algorithm cannot be guaranteed to deliver the correct result each time that it is executed

Let me warn you, however, that the chances of finding an efficient program are incredibly small. Very talented researchers all over the world have been attempting to do it for hundreds of problems, and nothing has yet emerged. Many of these researchers have intellects the size of the known universe, but have yet to come anywhere near finding an efficient algorithm for an NP-complete problem.

Win a Nobel Prize

If you are interested in winning a Nobel prize, the following step-by-step description tells you what you need do.

- Find a convenient NP-complete solution. The travelling salesman problems would do. It's reasonably easy to program and also has major applications in operations research.
- Develop an algorithm that you think will solve the problem in an efficient time. Write a program which implements the algorithm.
- Write a brute-force program that solves the problem.
- Run both the 'efficient' program and the brute force program using an increasingly large data set.
- Time each run. A stop watch would do for this.
- Plot each time against the size of the data set.
- If the graph for your algorithm looks like (a) in Figure 3 below, then congratulations, it's time to ring up the Nobel prize committee. Unfortunately, your graph will almost invariably look like (b).

This process can be started on a common-or-garden microcomputer. Usually, the graph will quickly tell you whether your efficient program is really efficient. If the curve for your program is still rising slowly when the size of the data set blows the micro, transfer your program to a mainframe. Examine its behaviour again. If the graph for the efficient program only rises comparatively slowly, then there is a high probability that you have cracked one of the great unsolved problems in computer science.

When researchers started work on algorithmic complexity, the main charge levelled against it was that it was hopelessly theoretical, with no possible application in computing. Even 20 years ago, there seemed to be rather Philistine approach to 'blue skies' research, although this attitude has strengthened recently. However, it did not take long for computer scientists to discover useful applications of NP-complete problems. Probably the best known is in cryptography.

Cryptography is the study of secret codes: their invention, transmission and decoding. Since the massive increase in electronic communication over the last two decades, interest in cryptography has blossomed. Most of the research in this area is being carried out, in secret, by security agencies, banks and computer manufacturers.

Someone who needs secure communication must take his message and apply some transformation to it. For simplicity's sake, let us assume that the message consists of two digit numbers, and the transformation t - not a very secure one - is that of adding an integer x to each number sent. The message is now delivered via some communication line. A sender then decodes the message by applying an inverse transformation q. In my example, the inverse is to subtract x from each number. The integer x, the key, will have been sent separately to the receiver, enabling him to decode the message. For communication to take place, the transformations must have the property that when they are applied consecutively to a message the message remains unaltered. Thus, in my example, adding an integer x to an integer and then subtracting x gives the original integer.

This is one of the main planks of cryptography. However, the problem with the technique I have described is that it is easy to crack the code without the key. If the integers are the internal codes of characters in a natural language text, the message can be easily broken by recourse to standard tables which give the frequency of letters in a natural language text. Sophisticated methods now exist which can break such codes.

Rivest, Shamir and Adleman, three American researchers, devised a cryptography scheme which, if an intruder wished to crack a code without knowing the key, would require them to solve an NP-complete problem. In practice, the problem normally used for this purpose is the factorisation of a very large number (containing more than 200 digits). So far, all attempts at devising methods to crack these codes have failed. Either the new algorithm turned out to be flawed or, if it worked, then it relied on the factorisation of a very large number... a problem that has been already shown to be grossly NP-complete.

Tinpot brains

I find the existence of NP-complete problems quite reassuring. Too often we believe that the computer is omnipotent, and can solve any problem that is given to it. This is often the basis for much of the optimism about artificial intelligence, and the predicted transfer of many of the tasks that we are good at to computers. The existence of NP-complete problems gives us much hope that human flexibility and built-in heuristics will enable us to outwit any computer in complex tasks.

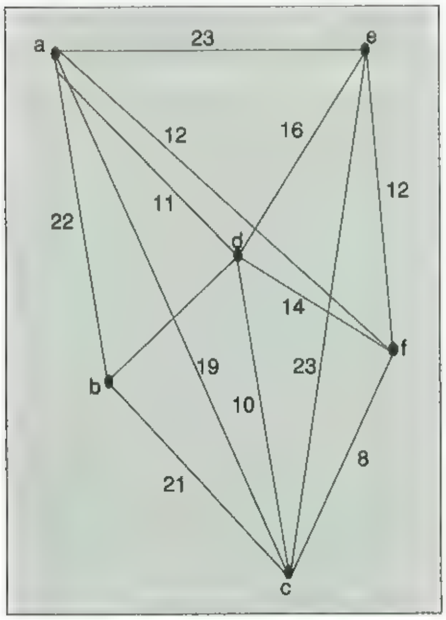


Figure 2 - The travelling salesman problem.

It would be tempting to finish this article on such this upbeat note. However, there is a new strand of research which is attempting to beat NP-completeness. Researchers in this area have developed 'algorithms' which give rise to efficient programs for NP-complete problems. Now this may confuse you because, earlier on, I stated that it was highly unlikely that such algorithms exist. However, the researchers have overturned the idea of an algorithm in their search for efficiency.

Congratulations, it's now time to ring up the Nobel prize committee with your news

Look up the definition of an algorithm in any computer science book, and what you will find is a variant of this statement: 'an algorithm is a step-by-step procedure for solving a problem which is guaranteed to give the same solution, for the same data, each time that it is executed'. Researchers have developed a new type of algorithm called a *probablistic algorithm.* The main feature of such an algorithm is that it will be fast, exceptionally fast even when applied to an NP-complete problem, but that in contrast to a conventional algorithm, it cannot be guaranteed to deliver the same, correct result each time that it is executed.

This doesn't seem to be very helpful. However, probablistic algorithms have been devised where the probability of delivering an incorrect answer, although finite, is very low indeed. If the probability of getting an incorrect result is *lower* than an incorrect result being delivered due to a circuit malfunction in the computer used to execute the algorithm, then you can see that the technique can be most useful.

Here is an example. Consider the problem of selecting one integer from a set of 1,000,000 (randomly ordered) integers, such that the chosen item lies in the top half of the set in terms of size. To do this conventionally, we might develop a program to find the maximum integer from the set. This guarantees the result, but requires that we thrash through the whole lot, performing something like 999,999 (1,000,000-1) comparisons. An obvious refinement is to stop the maximum program when it has examined over half the integers; this requires 500,000 (1,000,000/2) comparisons, and seems to be the optimum solution, *if we insist that a correct result is always given.*

A probablistic algorithm to solve this problem would select k numbers from the set and then find the maximum of these k numbers. If k is 20 then the probability of giving a correct answer is .9999999. If k is set at 100, the probability of things going wrong is negligible - much lower than the probability of the computer breaking down, an earthquake happening or you having a heart attack during the execution of the program. So we have an algorithm which requires only 100 operations, compared with 500,000 operations for a conventional algorithm, and which almost invariably delivers the correct result.

Domestic appliance

Probablistic algorithms are not only used for solving NP-complete problems. They are often employed in conventional software development. One of the most common examples is the concept known as *fingerprinting.* Consider a search of a large database of employee records, where each record consists of an employee's name and details. Programs accessing such a database will spend a lot of time comparing strings of characters to find a name match. Such comparisons can be quite time-consuming if the strings are very large: each character of one string has to be compared with a character of another string. To speed up the comparison, each name string is associated with a special integer, calculated when the

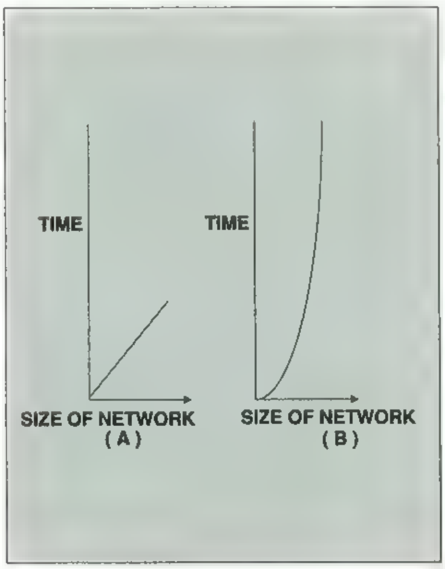


Figure 3 - Desirable and realistic growth of travelling salesman problem.

string is entered in the database, and stored with it in the record.

A fingerprinting subroutine is used to calculate the integer, the only proviso being that such a subroutine guarantees that two different character strings are most unlikely to generate the same integer. A possible fingerprinting function might work like this: add up the internal codes of a string, and then form the remainder when the sum is divided by another (large) integer, where the divisor is smaller than the word size of the machine being used. To use the database first search for a record by its 'fingerprint' key - which can be done much faster than string comparisons. When an integer match is found, you must then perform a full string comparison, character by character, to check that the strings are equal.

Genetic Engineering

Fingerprinting is currently one of the hot techniques in genetic engineering computing. Genetic engineering scientists are currently trying to unravel the genetic code: the sequence of chemicals contained in the DNA molecule which determine whether we are going to be tall, have blue eyes or be susceptible to cancer. In order to do this, they have to compare long strands of DNA, stored on a computer as billions of characters. A computationally very exhausting problem. One of the most promising techniques for bringing the run-time of their programs down to the point where they may be guaranteed success, is the use of probablistic algorithms and, in particular, fingerprinting.

There is a tendency to think of computers as omnipotent

Algorithmic complexity and probablistic algorithms form one of the most exciting research areas in computing. Unfortunately, most current research is being done in secret. This not only occurs in cryptography, but also in computational genetic engineering, where the financial gains of, say, engineering a new strain of vegetable can be massive.

However, research into algorithmic complexity and probablistic algorithms has the attractive property that it is an area where the amateur, with a microcomputer, can participate, and where the disadvantage of being an amateur, and having limited computing equipment (even a humble PC) is much less than in most other areas of computing research.

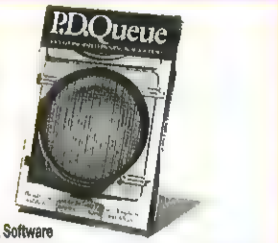
EXE

Darrell Ince is a Professor of Computing Science at the Open University. Until recently, he was acting head of the computing department.

OU WON'T FIND A FTWARE ANYWHERE*

MS Pascal Compiler	159.00
MS Powerpoint/Windows	295.00
MS Project/Windows	349.00
MS Quick C	59.00
MS Quick Pascal	67.00
MS QuickBASIC	59.00
MS Windows v3	75.00
MS Windows SDK	295.00
MS Word 5	259.00
MS Word 5/LAN	849.00
MS Word/Windows	299.00
MS Works	99.00
MSDOS 3.x w/GWBASIC	79.00
MSDOS 4.x w/GWBASIC	89.00
Multi-Edit Professional	129.00
Multimate Advantage II	289.00
Multimate 4	299.00
Netlib Library	149.00
NetUtils	63.00
NeuroShell	139.00
NewsMaster II	59.00
Norton Utilities	49.00
Norton Advanced	99.00
Norton Backup	79.00
Norton Commander	79.00
Norton Editor	49.00
Norton Guides C, OS/2 etc	59.00
NoSquint	45.00
Nutshell Plus	189.00
OmniPage/386	529.00
Opt-Tech SORT	125.00
Option Board Deluxe	105.00
Optune	59.00
OrCAD/SDT	495.00
Org Plus Advanced	85.00
Overlay() for C/Clipper	72.00
P.D.Queue for 1-2-3	59.00
PageGarden	63.00
Pagemaker for DOS	479.00
Pagemaker/Mac	569.00
Paradox 3	375.00
Paradox C Library	295.00
Pathminder Plus	79.00
PC Anywhere	75.00
PC Paintbrush IV Plus	155.00
PC Tools Deluxe	79.00
PC Write	79.00
PC-Browse	49.00
PC-File 5	79.00
PC-Fullbak Plus	69.00
PC-Globe	69.00
PC-Hookup	79.00
PC-Kwik Power Disk	59.00
PC-Kwik Power Pack	83.00

PC-Mix	49.00
PDC PROLOG for DOS	199.00
PerFORM	179.00
Personal Measure	59.00
Perspective Junior	129.00
Pertmaster Advance	1649.00
PFS:First Choice	129.00
PFS:First Graphics	129.00
PFS:First Publisher	129.00
Pixie Graphics	169.00
Pizazz Plus	79.00
PKZIP/PKUNZIP	54.00
PolyMake	110.00
PolyShell	79.00
Popdrop Plus	75.00
Power BASIC	69.00
PowerMenu	69.00
PreScript Deluxe	299.00
Primetime Personal	75.00
PrintAPlot	179.00
Printcache	95.00
Printmaster Plus	49.00
PrintQ	89.00
Printrite	85.00



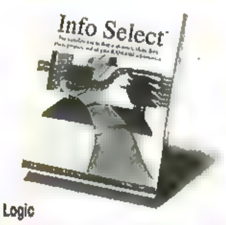
Funk Software

P.D.Queue v1.0 - The new add-in print spooler that lets you work in 1-2-3 or Symphony while you're printing. No more waiting around for your printer. The perfect companion to Sideways and Allways •••••• £59

Procomm Plus	72.00
Procomm Plus w/Viewdata	107.00
Professional File	199.00
Professional Write	199.00
Professor DOS	52.00
Project Scheduler	369.00
Project Workbench	795.00
Prokey Plus	79.00
PVCS Professional	379.00
Q&A	225.00
QDOS-II	49.00
QEdit TSR	99.00

QEMM/386 w/Manifest	59.00
Qram w/Manifest	57.00
Quattro Pro	289.00
Quick Schedule Plus	59.00
Quicksilver Compiler	359.00
R&R Report Writer	135.00
R&R Clipper/Fox Module	49.00
R:Base	375.00
Rapidfile	195.00
ReadRight OCR	399.00
Reflection 4+	225.00
Reflex	179.00
Relay Gold	179.00
Remote 2 Complete	129.00
RM/FORTRAN	379.00
Sage Professional Editor	149.00
SCO XENIX/386 Complete	1275.00
Seemore 1-2-3	65.00
Service Diagnostics Kit	395.00
Show Partner F/X	245.00
ShowText	149.00
Sidekick Plus	99.00
Sidetalk	79.00
Sideways	39.00
Silvercomm Library	139.00
SitBack	62.00
SiteLock w/Anti-Virus	329.00
Smalltalk V/286	145.00
Smalltalk V/PM	325.00
Smarterm 320	125.00
Smartware II	389.00
Soft Kicker Plus	95.00
Soft Type/Windows	129.00
Software Bridge	85.00
Software Carousel	57.00
Sourcer/486 w/BIOS	129.00
SPF/PC	155.00
Spinrite II	69.00
Sprint	139.00
SPSS/PC+	599.00
StatGraphics	429.00
Sterling Accountant Plus	279.00
Sterling Financial Controller	379.00
Super PC-Kwik	53.00
Superbase 4/Windows	449.00
Supercalc 5	195.00
Supercalc 5/LAN	469.00
Superkey	57.00
SuperPrint/Windows	149.00
SuperProject Expert	459.00
Switch-It	63.00
Systat w/Graphics	599.00
System Sleuth	85.00
Timeline	469.00

Timeworks DTP	85.00
Tornado	59.00
Total Recall	79.00
Turbo C++	129.00
Turbo C++ Professional	199.00
Turbo Debugger & Tools	99.00
Turbo Pascal	79.00
Turbo Pascal Professional	149.00
TurboCAD	129.00
Typing Tutor IV Plus	39.00
Typografica Prime	49.00
UI Programmer 2	295.00
UI Touch & Go	235.00
UltraScript PC Plus	245.00
Ultravision	79.00
Vcache	45.00
Ventura Publisher	495.00
Ventura Pro Extension	269.00
Virex-PC	79.00
ViruSafe	79.00
VM/386	145.00
Vopt	45.00
Watchdog	185.00
Where in Europe	32.00
Who-What-When	139.00
Word for Word Pro	99.00
WordPerfect	269.00
WordPerfect Addtn'l Users	149.00
WordPerfect Library	82.00
Wordstar 2000+	237.00
Wordstar Pro	205.00
Xtree Pro Gold	89.00
XYwrite III Plus	269.00
Zortech C	59.00
Zortech C Video Course	179.00
Zortech C++ Developer	269.00
Zstem-240	199.00



Micro Logic

Info Select v1.0 - The superfast, exciting new way to manage notes, lists, ideas, plans, contacts and all your RANDOM information. Memory-Resident. Easy-to-use yet powerful. Endless uses ••••••••••• £85

LY ADVERTISED PRICE+

UR POLICY

- All prices and manufacturer's specifications are subject to change without notice.
- All prices exclude carriage and VAT.
- All orders subject to availability.
- Please specify media size when ordering.
- Goods are not offered on a trial basis.
- Goods are offered for sale subject to our standard conditions of sale which are available on request. E&OE.

* Provide written proof of any U.K mail-order software company who supplies a wider range of IBM PC software than we offer in our catalogue within 14 days of this advertisment and get yourself a FREE copy of PC Tools Deluxe!

+ We are always delighted to be given the opportunity to beat other nationally advertised prices (at the time of placing an order) subject to the same terms and conditions of the company being beaten.

All Trademarks and Registered Trademarks are hereby acknowledged.

VISA

SOFTWARE PARADISE

Leading the Way

Software Paradise, Avenue House, King Edward Avenue, Caerphilly, Mid Glamorgan, CF8 1HE

FREEPOST
—TODAY—
NO STAMP NEEDED

☐ Please send me my *FREE* copy of ***'The Discerning Person's Guide to Software'***

NAME

COMPANY

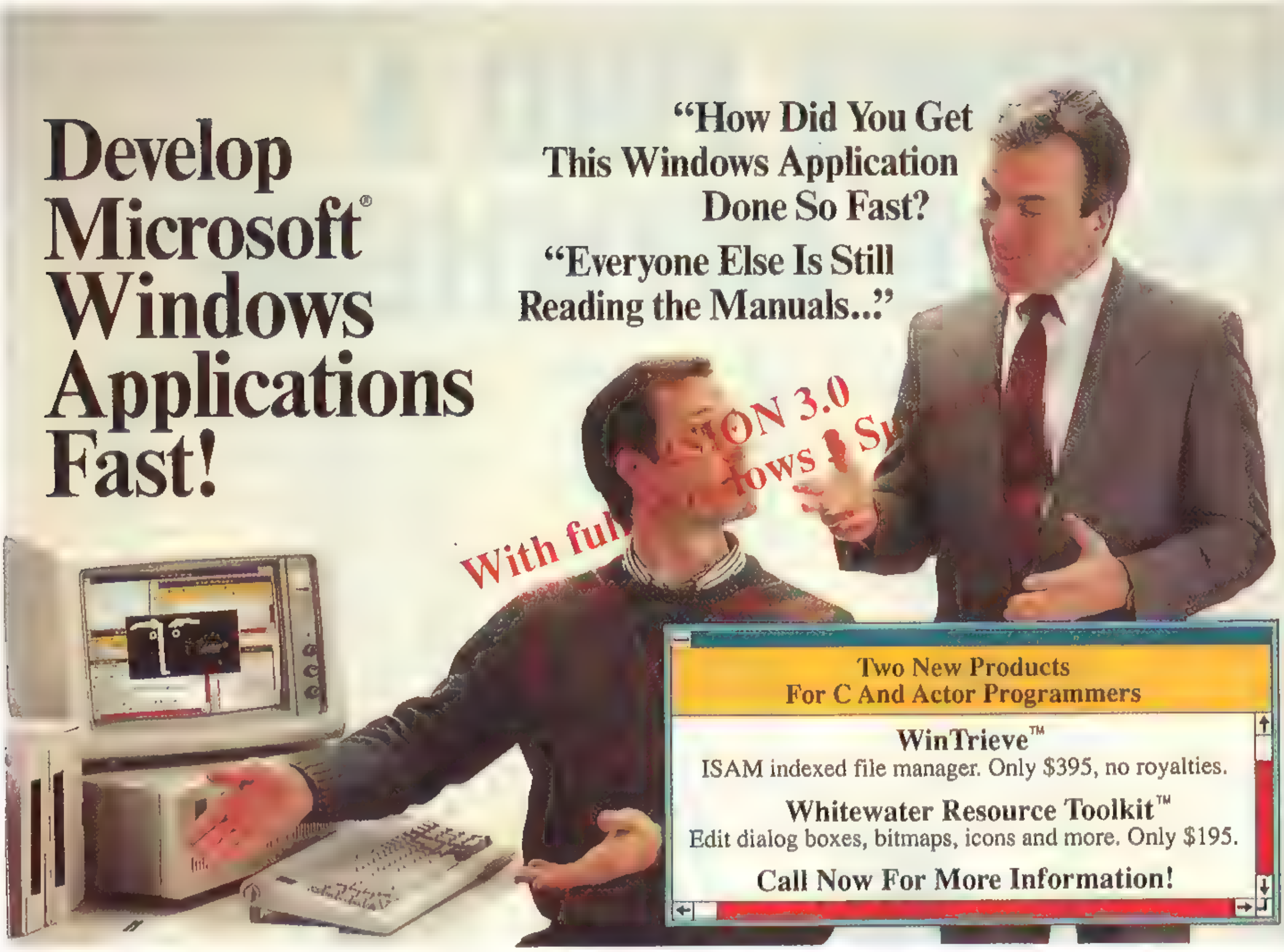
ADDRESS

POSTCODE TEL

Post now to: Software Paradise, FREEPOST, Avenue House, King Edward Avenue, Caerphilly, CF8 1ZZ

• EXE 10.90

CIRCLE NO. 288



The Windows Advantage

You've selected Microsoft Windows as your graphical user interface. Windows applications are easier for end-users to learn and use. It is **the** standard for PCs.

Now you want Windows applications completed as fast as possible. But it takes time for your programmers to get up to speed on the intricacies of Windows development.

Why The Delay For Windows Applications?

Programming for the Windows environment is very different from character-based programming. Just learning the 450 Windows function calls can take months. Getting used to its event-driven system and memory management takes additional time.

Programmers who are new to Windows development are looking for support to help them produce high-quality Windows programs quickly and efficiently.

3 Ways To Speed Up Windows Application Development

Now there are specific ways to speed up applications development under Windows.

- **Code reduction.** There is a fast, effective way to develop Windows applications. A way that cuts the amount of code needed. A way that by-passes much of the complexity of Windows programming. A way that doesn't compromise performance.
- **Short learning curve.** There is an easy, convenient way to get your entire programming staff up to speed with Windows within weeks.
- **From prototype to complete application.** There is a way to quickly develop prototypes of Windows applications in days, not months. Then, turn those into completed applications quickly.

1 Product, 3 Solutions

There is **one** product which brings **all three** solutions to your Windows programmers. It's called **Actor**®.

Actor is a proven product. It's now being used in thousands of software development groups world-wide.

Actor Does Windows

Actor gives your programmers a fast, effective way to develop Windows applications. **Actor's** complete development environment allows you to produce everything from prototypes through stand-alone applications. Programs developed in **Actor** are fast and support all Windows features.

Actor offers interactive source code debugging and testing—greatly reducing development time. Your programmers benefit from high code reusability because **Actor** is an object-oriented programming language. This allows them to build on existing code, eliminating duplication of effort. They can even dynamically link to existing C code.

Actor also provides one more essential: fast, responsive support from The Whitewater Group®, recognized worldwide as leaders in Windows programming.

If you are programming applications for Windows, you should be using **Actor.** Users report that **Actor** cuts Windows programming time by at least 50%.

Act Now!

See for yourself how **Actor** can help your Windows programmers! The first step is to order our free booklet, **For Faster, Better, Smaller, Smarter Windows Programming.** Examine detailed information about using **Actor,** complete with actual source code examples.

Call us at 312/328-3800. We'll send you the booklet and complete details about **Actor.**

Take 60 seconds now to save months of development time in the future.

NEOW **- The Windows Specialists**
Unit 12, Progress Business Centre,
Whittle Parkway, Slough. SL1 6DQ.
Telephone: 06286 68334. Fax: 06286 65722

CIRCLE NO. 289

Top of the Pops

POP-11 is a powerful procedural language with much of the functionality of Common LISP and an object-oriented extension. Clark Morton explains its appeal.

The 'POP' in POP-11 is officially an acronym for 'Package for On-line Programming'. However, most people agree that it is derived from the surname of its originator, Robin Popplestone. If you ask Robin, he claims that the name was chosen because he was unlucky enough to be out of the room at the time. POP was originally created back in 1968, and has been developing ever since. The '11' suffix, incidentally, serves as a version number, but also shows the influence of the DEC PDP-11 computer (successive versions were named POP, POP-2, POP-10, WONDERPOP and POP-11).

Although POP-11 is a procedural language with an ALGOL-like syntax, it boasts functionality equivalent to the best modern LISP implementations. It is a general-purpose, symbolic processing language with specialist facilities such as structured pattern-matching and compiler-writing tools. POP-11 has been used in a wide variety of application areas including fault-diagnosis, process control, CAD/CAM, image processing, intelligent trading and compiler building.

To show how easy it is to use POP-11, Figure 1 shows a sample solution to the triangle problem. This direct approach does not take advantage of any of POP-11's more powerful features. There are, however, one or two points of interest. Note that assignment (represented by the assignment arrow `->`) goes from left to right, and that data can be printed by using the print arrow =>.

Dynamic Typing

POP-11 identifiers are dynamically typed. Type information is associated with the value of the identifier, rather than with the identifier itself. Consequently, when declaring identifiers, it is not necessary to specify the type.

A common misconception which follows from this is that POP-11 is weakly typed. Not so. Type-checking is strong, but is generally performed at run-time rather than compile-time. The checking is done using *recogniser* procedures that exist for each type. This makes it easy to define procedures that behave differently depending on the type of their arguments. For instance, the built-in routine `checkinteger` takes three arguments: the object to be checked and an integer lower and upper bound. However, the second and third arguments may be set to `false` when there is no lower and/or upper bound.

Shifting the burden of type checking from compile-time to run-time is a design trade-off: you lose execution speed, but gain flexibility. As a bonus, POP lets you have your cake and eat it - operations also have faster, non-type checking equivalents.

Figure 1 - Simple Triangle solution.

```
define is_triangle(triangle);
lvars triangle;
    if triangle(1) <= triangle(2) + triangle(3) and
       triangle(2) <= triangle(1) + triangle(3) and
       triangle(3) <= triangle(2) + triangle(1) then
       return(true);
    else
       return(false);
    endif;
enddefine;

define is_equilateral(triangle);
lvars triangle;
    if triangle(1) = triangle(2) and triangle(2) = triangle(3)
    then
       return(true);
    else
       return(false);
    endif
enddefine;

define is_isosceles(triangle);
lvars triangle;
    if triangle(1) = triangle(2) or
       triangle(2) = triangle(3) or
       triangle(3) = triangle(1)
    then
        return(true);
    else
        return(false);
    endif
enddefine;
```

```
define triangle_problem();
lvars triangle i;
    newarray([1 3]) -> triangle ;
    ;;; build an array by using the built-in procedure newarray
    ;;; it takes a list as its arg to specify the "index range"

    for i to 3
    do
        pr('Enter the size of side ' >< i >< ' please ##')
        ;;; the >< is used to concatenate strings
        itemread () -> triangle(i);
        ;;; itemread just gets the next item from the keyboard
    endfor;

    if is_triangle(triangle)
    then
        if is_equilateral(triangle)
        then
            'This is an equilateral triangle' =>
        elseif  is_isosceles(triangle)
        then
            'This is an isosceles triangle' =>
        else
            'This is a scalene triangle' =>
        endif;
    else
         'This is not a triangle' =>
    endif;
enddefine;
```

The Open Stack

POP-11 provides explicit access to the argument stack. While it is possible to write programs that make no use of this, there is no doubt that it can add to flexibility and efficiency, and allows programmers to code in a functional style. It is easier to understand some of the language's more advanced features in terms of stack operations.

Here is the 'stack view' of an assignment, which breaks into two operations: (a) leave the result of the evaluation of the expression on the left hand side of the assignment arrow on the stack and (b) pop the topmost item from the stack into the variable on the right hand side of the assignment arrow. Building on this idea, here are three useful special cases of the POP-11 statement.

```
100*10;
```

leaves 1000 on the stack (part (a) above),

```
-> fred;
```

pops the object on top of the stack into fred (part (b) above), and

```
100 * 10 ->;
```

carries out the evaluation of the expression but then erases the result from the stack.

The open stack allows procedures to return multiple results in a natural way. The following procedure returns two results:

```
define foo();
100;
200;
enddefine;
```

Incidentally, the arguments to parameterised procedures are bound at runtime top-first against right-most.

List Processing

POP-11 provides powerful support for the creation and manipulation of lists, with functionality familiar to Common LISP users. Lists are constructed by specifying data within square brackets:

```
[1 2 3] -> fred;
```

Various special symbols are recognised within lists.

```
[^a] -> fred;
```

splices the value of the identifier a into the list and

```
[^^a] -> fred;
```

assumes that the value of the identifier a is a list and splices each top-level object into the list. There are many other useful operations of this type.

Decorated brackets are employed in list and vector construction. The contents of the decoration are evaluated as a piece of POP-11; any results left on the stack are spliced into the list or vector. For example:

```
[1 2 3 4 5 %sqrt(36)%] -> fred;
fred =>
** [1 2 3 4 5 6]
```

This feature can be employed to make list and vector construction more explicit.

Return to Triangle

Figure 2 shows a reworking of the triangle problem, written as a POP-11 programmer might do it. First note that the `is_` procedures have changed slightly; results are not explicitly returned. The body of each procedure is a single expression. The result of evaluating this expression is left on the stack and is used as the result of the procedure. These are purely functional versions. Unnecessary stacking and unstacking is avoided.

Second, the equality operator = has been replaced with ==. The = operator always compares each part of the operand. The == operator behaves the same way as = for simple comparisons of scalar entities (so its use in Figure 2 makes no difference to the object code). However, when multi-component objects (such as strings, arrays or lists) are tested, the operator compares only the address of the operands. This is much more efficient.

```
define is_triangle(triangle);
lvars triangle;
    triangle(1) <= triangle(2) + triangle(3) and
    triangle(2) <= triangle(1) + triangle(3) and
    triangle(3) <= triangle(2) + triangle(1)
enddefine;

define is_equilateral(triangle);
lvars triangle;
   triangle(1) == triangle(2) and triangle(2) == triangle(3)

enddefine;

define is_isosceles(triangle);
lvars triangle;
    triangle(1) == triangle(2) or
    triangle(2) == triangle(3) or
    triangle(3) == triangle(1)
enddefine;
```

```
define triangle_problem();
lvars triangle;

    pr('Enter the sizes of sides please ##');
    [% dl(readline()) %] -> triangle;

    if is_triangle(triangle)
    then
        if is_equilateral(triangle)
        then
            'This is an equilateral triangle' =>
        elseif is_isosceles(triangle)
        then
            'This is an isosceles triangle' =>
        else
            'This is an scalene triangle' =>
        endif;
    else
        'This is not a triangle' =>
    endif;
enddefine;
```

Figure 2 - POP-11 approach to Triangle problem.

Figure 3 - Accessors and updaters.

```
Using two identifiers to get at salary data:

get_salary("john") -> johns_salary;
put_salary(2 * johns_salary,"john");

Using one identifier, with accessor and updater, to get at salary data:

2 * salary("john") -> salary("john");
```

Finally, to get the sizes of the sides, the built-in procedures `dl` and `readline` are used. `readline` returns the next line from the keyboard as a list, `dl` leaves the top level contents of a list on the stack.

Other Data Types

POP-11 supports a similar range of types to Common LISP, including big integers, complex numbers, ratios, association tables, vectors and arrays. It is easy for users to define new types.

Procedures are treated as first class data objects, and can be passed as arguments, returned as results, spliced into lists and generally slotted into any data structure. POP-11 procedures can be created dynamically:

```
define foo();
    procedure();
        'hello' =>
    endprocedure;
enddefine;
```

The result of calling `foo` is a procedure. When this result is run, it prints hello:

```
foo() -> fred;
fred();
** hello
```

This procedure syntax used allows the creation of 'un-named' or anonymous procedures.

Most procedures appear on the left hand side of the assignment arrow as part of an expression. However, it's important to realise that procedures are no more than identifiers whose value is a procedure. In POP-11, such identifiers can appear on the right hand side of the assignment arrow as well. For instance, the built-in identifier `hd` accesses the head of a list when used on the left hand side of an assignment. Placed on the right hand side, it updates the head of a list.

```
[1 2 3] -> fred;
100 -> hd(fred);
fred =>
** [100 2 3]
```

In fact, the identifier `hd` has **two** procedures associated with it; an *accessor* which is used when `hd` is on the left hand side and an *updater* which is used on the right hand side. This facility is ideal for defining procedures that behave differently depending on the context. For instance, you *could* write two procedures, called `get_salary` and `put_salary`, to access a salary database. However, it is much clearer to use a single identifier `salary`, and write an accessor and updater for it (see Figure 3).

A *closure* is a procedure that is produced when another procedure has some of its arguments 'pre-bound'. For instance, to build a procedure that checks that its argument is an integer, you could 'freeze' as false the second and third arguments of the built-in procedure `checkinteger`:

```
checkinteger(%false,false%)
                -> testinteger;
```

Closures have their arguments frozen from the right. They are useful for modifying general procedures to specific needs, and also when there is a requirement for procedures to 'carry around' some data with them - cf C++'s initialised arguments.

Oddments

The pattern matcher is a powerful tool for matching templates against lists. The template can contain a number of special symbols allowing the non-determinstic binding of the list against the template.

```
[1 2 [3]] matches [??x 2 [3]]
```

would return true, and the identifier `x` would bind against 1.

Flavours is an object-oriented extension to POP-11. It provides multiple inheritance, a set of powerful standard methods, user definition of methods and the use of 'meta-flavours'. Flavours is itself implemented in POP-11 and supplied as source (and is thus a good example of the extensibility of POP-11's syntax). Following naming trends in object-oriented languages, the combination of POP-11 plus Flavours is known as 'POP++'!

Figure 4 shows the definition of three 'flavours' (confusingly, the term 'flavours' applies both to the language extension and to its main component). Flavours are 'object templates' (or 'classes'). To create a poly-

```
flavour polygon;
ivars sides;
lvars side total;
    defmethod printself;
        pr('<');
        for side in sides
        do
            pr(side); pr(' ');
        endfor;
        npr('>');
    enddefmethod;
    defmethod perimeter;
        0 -> total;
        for side in sides
        do
            side + total -> total;
        endfor;
        total;
     enddefmethod;
endflavour;
```

```
flavour triangle isa polygon;
    defmethod before printself;
        pr(' Triangle ');
    enddefmethod;
    defmethod after initialise;
        unless is_triangle(sides)
        then
            mishap('This is not a triangle',sides);
        endunless;
    enddefmethod;
endflavour;

flavour equilateral_triangle isa triangle;
    defmethod before printself;
        pr(' Equilateral');
    enddefmethod;
    defmethod after initialise;
        unless is_equilateral(sides)
        then
                    mishap('This  is  a  not  a  equilateral
triangle',sides);
        endunless;
    enddefmethod;
endflavour;
```

Figure 4 - An object-orietated solution to Triangle problem.

Figure 5 - Using object instances.

```
make_instance([polygon sides [1 2 3 4]]) -> fred;
make_instance([equilateral_triangle sides [2 2 2]) <- printself;
Equilateral Triangle <2 2 2>
```

gon object (an *instance* in OOPS parlance) we use the `make_instance` call, as shown in Figure 5. Each flavour has slots (instance variables for holding data) and methods (procedures that act on the instances). Slots are specified by using `ivars` declarations.

As expected, flavours can inherit methods and slots from their parents. In the example, `triangle` inherits from `polygon`. In a sense, `equilateral_triangle` is the 'most specific' flavour and `polygon` the 'least specific'. Methods in child flavours replace or 'shadow' methods in their parents with the same name. When a method is run, all of the 'before methods' are run in most-specific first order, then the method itself is run, then finally all of the 'after methods' in least-specific first order. Methods are 'fired' using a special assignment arrow, see Figure 5. The analogy of 'sending a message' is often used when firing methods.

Availability

The dominant implementation of POP-11 is available within the POPLOG program development environment. POPLOG is an integrated environment containing incremental compilers for POP-11, Prolog, Common LISP and ML. POPLOG is available on Sun workstations, VAXs (VMS and ULTRIX), DECstations, Apollos and HP 9000s; prices for the package start at around £7,500 (Sun version). An implementation on the PC under UNIX has been developed recently, but is not commercially available at the time of writing. POPLOG is developed and maintained by the University of Sussex. A subset of POP-11 called Alpha-POP is available for the Mac.

The POPLOG and POP Languages User Group is an independent user group set up to promote the use of POPLOG and POP-11. It may be contacted via Integral Solutions Ltd (phone number given below).

Like many other computer languages, POP is undergoing standardisation. The BSI is currently sponsoring an initiative aimed at producing a standard for POP-11 by 1992. This work is strongly supported by the implementors, vendors and users of POP-11.

EXE

Clark Morton has worked with POP-11 and POPLOG since joining SD-Scicon in 1984. In 1989 he helped to set up Integral Solutions Limited (0256 882028). ISL markets POPLOG on behalf of Sussex University.

If you are interested in reading more about POP-11, then he recommends the following titles: POP-11: A Practical Language For Artificial Intelligence *(by R Barrett, A Ramsey and A Sloman, pub Ellis Horwood, ISBN 0-853112-924-X),* Programming In POP-11 *(by Jonathan Laventhol, pub Blackwell Scientific Publications, ISBN 0-632-01528-4),* Computers And Thought *(by Mike Sharples, David Hogg, Chris Hutchison,* Steve Torrance *and David Young, pub Bradford Books/The MIT Press, ISBN 0-262-19285-3) and* POP-11 Comes Of Age *(Edited by James A D W Anderson, pub Ellis Horwood, ISBN 0-7458-0680-5).*

login: fred password: XXXX

UNIX is a secure system - if the people who use it choose sensible passwords. Peter Collinson explains the "dos and don'ts" of password selection.

In the last few weeks I have taught a couple of courses on UNIX fundamentals and System administration. The students on both courses were worried about tales that they had heard, tales portraying UNIX as a very insecure system. Since I have been involved with running UNIX in the hostile University environment for 15 years or so, I believe that UNIX is reasonably secure and I cannot see why people are so worried about possible problems. I suppose the folklore has circulated because UNIX has suffered from two well-publicised breaks in security in the last couple of years.

First there was the 'Internet Worm', a program that escaped and rampaged around the US national network. The program exploited some holes in several programs to bootstrap itself from machine to machine. Once on a machine, it set about finding other machines to infect using a variety of techniques. It should be said that most of the holes exposed by the worm have simple fixes and most manufacturers have released fixed software.

Second, there were the various attacks on machines described in Clifford Stoll's book *The Cuckoo's Egg*. These attacks seemed to emanate from the German X.25 network. Here, the original invasion of the machine was made with the unwitting help of a legitimate user. The user had a trivial password that was easy to crack or guess. Once on the machine, the intruder used a set of different methods to circumvent the security system.

So, security on UNIX depends on passwords, and passwords depend on people. The people selecting bad passwords are the real problem and not UNIX *per se*. The rules for choosing better than trivial passwords have been known for some time; getting people to accept these rules is difficult. This article will show you how easy it is to crack simple passwords; and suggest how you should invent passwords that are harder to guess.

Password system

Passwords are stored in the file `/etc/passwd` along with other per-user data. The password is encrypted using the `passwd` program when the user selects it, and is written into the password file in this form. When the user logs into the system, the program asks for his password. The string typed in by the user is encrypted, and the resulting string is compared with the one in the password file. The C looks something like:

```
/* use getpass to turn off */
/* terminal echo */
/* new is a char* */
new = getpass();
/* get encrypted string */
/* cs is a char* */
cs = crypt(new, pw->pw_name);
/* and compare */
if (strcmp(cs, pw->pw_passwd))
fail();
elsesuccess();
```

Notice that the password is not *decrypted*. This is why your system administrator gets you to supply a new password when you forget the old - having forgotten it, no-one knows the old password. The process of matching passwords is done by encryption, so the important criteria is that the passwords should always *encrypt* to the same string. The `crypt()` routine is lurking in your C library and anyone can use it to take a password and compare the result against the entry in the password file. It makes it easy to write a password cracker too - simply take various lists of words, `crypt()` them one at a time and compare the result against the password file entry.

The knowledge

Should anyone mind that I am telling you all this? No. No. No. The people who stand up and say 'my system is secure because no-one has the information needed to crack it' are living in some fairy land. Keeping knowledge from people is a poor substitute for real security. It just takes one persistent person to gain the knowledge, break the system and the walls all fall down. Security by obscurity simply doesn't work.

It is better that you know how the security system works, and when you are using a safe or unsafe password. You are then helping to protect the whole machine, since invasion of your files really means that the whole system is open for access by some intruder. It's important that all UNIX users are educated into the nuances of password security - since only then will your machine start being really secure.

None of this stuff is new, the knowledge has been around for some time. The original password cracking paper was written by Robert Morris and Ken Thompson in 1979 - and not much has altered since then. The main thing to realise is that the people attempting to hack into your system know all this stuff already - so you must know it as well.

Cracking passwords

As part of the course for Systems Administrators that I helped to run recently, I was allotted with the talk on 'Security'. So, thinking about all these issues, and as an exercise, I ran a public domain password cracking program on a password file containing 850 users. The idea was to compare the encrypted passwords against the 25,000 words in the standard word list `/usr/dict/words`. I didn't mess around trying to tune the program and the `crypt` algorithm. On a Sparc 1, I was processing each password in around 10 minutes elapsed time. This seemed a little slow, so I split the password file up into bits and ran the program in parallel on several machines. I got about 10% of the passwords. This is apparently typical.

Actually, the nice thing about this event was that I ran the password cracker on the passwords that the students had entered into the machine as part of the course; and got 10% of them too. Great lesson.

Dan Klein from Carnegie Mellon University has done even better. At the recent UKUUG conference held in London, he published some results from tests he has been doing

G.W. Computers Inc
ADCI, 4 Eagle Square,
E.Boston, MA02128
U.S.A.
Tel 617-569-5990
Fax 617-567-2981

System-IX (Networks) Ltd
55, Bedford Court Mansions
Bedford Avenue, London
WC1B 3AD U.K.
Tel 071-636-8210
Fax 071-255-1038

Menu-IX | System-IX | DBMS-IX | Sysix | AI | Spell | Merge | Lock | GURU | Crypt | Tawk | Tgrep

Cut · More · Term Ctl · Where · Sort · Grep · Idle · CB Trees · Evaluate · Awk · Banner · C Source · Fgrep · Help · Head · Tail

Xenix · Xenix Net · Unix · Interactive · System V · TCP/IP · DOS · Token-Ring · OS/2 · Ethernet · Novell Lan · Mic Net · Aix · VP-IX · VMS · QMS

MenuIX, SysIX & SystemIX under Xenix,Dos & Unix.

Main Options

Administration · Files · Backup · Maintenance · System · Acct · Mail

Ad Options
- C Shell
- Bourne Shell
- Visual Shell
- Shutdown
- Startup
- Install Applic
- Remove
- Applic
- Display
- Process
- Display TTY
- Modify TTY

File Options
- Free Space
- Maintenance
- Create Dir
- Remove Dir
- List Dir
- Print Dir
- Create File
- Remove File
- Print File

Back Options
- Backup Files
- Backup Filesystem
- List Backup
- Print Backup
- Restore Files
- Format Diskette
- Copy Diskette

Mnt Options
- User
- Printer

Invoice
Letters

Acct Options
- Account User
- Account Group
- Account TTY
- Account All
- Enable Account
- Disable Account

Mail Options
- Phone To User
- Send Mail
- Read Mail
- Write All
- UUCP

Print Options
- Add Printer
- Remove Printer
- Enable Printer
- Disable Printer
- Start Schedule
- Stop Schedule
- Lp Status
- Print File
- Set Default Lp

User Options
- Create User
- Remove User
- Create Group
- Remove Group
- Change Password
- Applications

£49.00

App Options
- Spreadsheet
- Wordprocessor
- Database
- Other

Db Options
- Sys-IX
- System-IX
- Guru
- Csh-IX

UUCP Options
- UUCP Setup
- Dialcode
- System
- Devices
- UUCP Stat
- Clear Log
- Remove Log

Sysix ......From **£99.00** to £295.00

Scrollable-resizable Help

Everthing onscreen is runtime interpreted from your own edited ascii file
Online Hot-Key dynamically scrolls moveable context help from your ascii file
Unlimited chaining through pull-down menues with ITEM status line help
Supports full system and multiple system calls, auto-minimise to 7k under DOS
256 colours, highlight bars, unavailable items. All formats XENIX,UNIX,DOS
Edit ascii file to translate entire runtime to any European language
Runs child system calls and script files. 120 page manual. Unbeatable value
Timeout feature for screen-blank and or password re-entry for networks

+++ Hardware Systems +++

From 386-16 SX w: 2mb ram/VGA/40mb-dsk £895.00
to 486-25 w: 4mb ram,VGA,.80mb disk £3750.00
call for quotations and latest details & prices.

SCO UNIX / XENIX

	286	386
SCO Xenix Op Sys	£357	£422
SCO Xenix Dev Sys	£357	£455
SCO Xenix Text Pro		£126
SCO Xenix Comp Sys	£775	£905
SCO Unix Op Sys		£525
SCO Unix Devel Sys		£613
SCO CGI Graph Run/Sys	>	£191
SCO Multiview	£256	£321
SCO VP/ix		£321
SCO Foxbase+	£516	£647
SCO Foxbase+ Run/Sys	>	£191
SCO Masterplan	>	£516
M/Soft Word v.5.0	>	£563
Multiplan	>	£321
SCO Eroff Documentor	£451	£516
SCO Lyrix v6.0	>	£386
SCO Professional	>	£386
SCO Image Builder	>	£386
M/soft BASIC compiler	>	£451
M/soft BASIC interp.	>	£256
M/soft FORTRAN comp.	>	£451
M/soft PASCAL comp.	>	£451
SCO XENIX-NET Server	£386	£451
SCO XENIX-NET 3 Server	£841	£1036
SCO TCP/IP (Rec streams) Run/Sys		£227

OPEN DESKTOP and XSIGHT also available, £call
Complete systems (runtime/dev) include MENUIX FREE.

INTERACTIVE 386/ix UNIX

New Rel 2.2		Application	Network	Workstation
Platform	(2 User)	£495	£695	£855
	(Multi)	£1095	£1350	£1495
Developer	(2 User)	£1095	£1155	£1295
	(Multi)	£1750	£1815	£1945

These are VERY powerful professional systems, For an Application starter you need about 12-20 mb of hard disk, but a workstation developer needs hard disk space to install 65 by 1.2 or 1.44 mb diskettes. To include things like : Op'sys, Soft'dev (LEX,YACC,SCCS etc), Standard C,NewC Text prc (NROFF,TROFF,TBL,etc), XII, XII Dev, Vpix (DOS SHELL), TCP/IP,NFS, Online MANS, Looking glass, Viewtan UI, Codewatch, Co-Edit. Full documentation weighs in at about 40 lbs. All modules can be purchased separately.

Some Parts	1-2 Users	Multi User
386/ix Runtime	£255	£495
Software Development System		£495
X11.3 Runtime	£240	£265
X11.3 Development System	£580	
Multiview		£395
Motif Development System	£365	
TCP/IP		£355
NFS	£355	£420
TCP/IP + NFS	£425	£585
PC Interface, DOS Bridge Module		£365

System kits over £900 include MENUIX, FREE.
System kits over £1500 include SYSIX and GURU FREE.

Utilities/Tools (Unix/Dos) $49.95 each.

G.U.R.U. Combines features of GREP and AWK found under UNIX. Enables exacting searches in records/windows of records, and the results processed / calculated / date-compared / and output formatted in unlimited ways.
SYSED-IX A powerful EMACS split-screen/colour editor that enables extensive customisation to particular user requirements. If you do not like VI, then this editor under Unix/Xenix has its own tutorial and help files making it easier to use.
CSHELL-IX A command processor under DOS that incorporates all the features of the Berkely UNIX C Shell relevant to dos.

Additional HOT Items

Q-Calc spreadsheet	£695
Master-Plan project management	£1295
Uniplex business software	£ 1150
PC-CONNECT	£450
SQL-Connect	£450
PC-TCP	£375
PC-Xview	£245
Xvision	£295
Coherent UNIX	£99
Designer C++	£1195
Terminal Control for UNIX AND DOS	£695
Menuix	£49.95
Sysix	£99.95
Sysix + Guru	£145
System-IX	£295
Sysed-IX	£295
Cshell-IX for DOS	£50

Unusual products : Entire Shakespeare works, Theasaurus & Dictionary, Bibles, 500,000 word lists, Moby Words,CD-ROMS, C sources, Microsoft, Borland, Lotus, Sage and most other well known brands. PLUS CD-ROM and WORM drives, Laser printers, (QMS,HP), and many well known brands

PORTABLE hard disk. FITS A STANDARD 5.25" DRIVE BAY. FRAME & REMOVEABLE DRAWER, takes a 3.5" drive of 40-200 mb into the draw, and remove it in 1 second, Just plug in any O/S of your choice, or take your files from machine to machine. JUST £55

Compaq | Toshiba | IBM | Epson | Qume | NEC | Seagate | Wyse | Relysis | Zenith | Priam | Ranger

CIRCLE NO. 296

using bigger dictionaries and special lists of words. In his sample of 15000 accounts from several sites in the US and the UK, he has cracked around 25% of all the passwords. This took him 12 CPU months. He adds '21% of the passwords had been guessed in the first CPU week, and in the first 15 minutes of testing 368 (or 2.7%) had been cracked'. So he used lots of CPU - but found some very quickly. This is all very worrying. He discovered that the most-used password in his sample is 'fred'; the second is a tie between 'password' and 'passwd'; the third is 'random'.

Foiling the cracker

Dan's work has shown that people simply do not think hard enough about the use of secure passwords. Users mostly choose passwords that are easy to crack. There is no reason for this - to create a sensible password means following a small set of rules that have been well understood for a decade or more.

First, there are some basic security rules that are often violated. Every account in the password file must have a password. The accounts in the machine that exist to provide 'dummy' users (`bin` or perhaps `sys`) should have a password or have the password field hand edited to star `'*'` - nothing encrypts to this.

Never have an account with no password calling some other program as a shell, however useful it is. I was at a site recently who had a `kermit` login with no password - 'it's safe because it changes into a public directory'. I pointed out that the `kermit` command has an 'escape to local shell' option - and using it, I could access the whole system. There is a way around this using `chroot` and I implemented this quickly.

You should never have logins called `guest` or `visitor` with no password or some easily guessed one. The Kent PSS gateway used to have a user of GUEST with a password of FRIEND - this is no longer the case, I hasten to add. People logging into VMS always try the standard DEC password for the SYSTEM manager or FIELD service, people logging into UNIX often try for an unguarded `guest` or `visitor` account.

When creating a password, don't use your login name in any form - forwards, backwards or capitalised. Also don't use your first and last name in any form; on many systems, your name is also stored in the password file and is easily available to the cracker. Don't use information readily obtainable about you: the names of your spouse, children, dogs, cats, car number plate, room number etc. Don't use a password of all digits, or all the same letter; this reduces the potential character set that the cracker has to use when searching for your password.

Don't use a word contained in dictionaries, spelling lists or other lists of words - this boils down to 'don't use a word'. If your name is Gwynfor, don't use a Welsh word - no doubt Welsh dictionaries exist on-line too. Dan had good success in testing the passwords of users with Chinese names against a Chinese dictionary. Don't use a password of less than six characters, Dan found that 35% of all passwords that he cracked in his sample were six characters - so use a password longer than that. Don't write the password down, this is just like leaving the key in the lock. Finally, sometimes it is tempting to give your password to others so they can easily get at your files - don't.

Do use a password that is easy to remember, so you don't have to write it down. Use a password with mixed-case alphabetics and with non-alpha characters - digits or punctuation. Make sure that you can type the password quickly, this makes it harder for people to guess your password if they watch you type it.

The best passwords are nonsense, but random nonsense can be hard to remember. One way round this is to take a phrase like 'Random rubbish is hard to remember' and create a password from its initial letters 'Rrihtr'. If you must use a word or full name, then complicate it by adding punctuation characters: 'zaphod' is in most crackers dictionaries by now, but it will be hard to find 'zap*hod'. Avoid putting the punctuation at the start or at the end - this is obvious.

Other methods

On balance, I am against generating passwords using some automatic system. If the password is random nonsense, it will be completely forgettable nonsense and so the user will commit it to paper (or the wall next to their terminal). One way of creating nonsense is to take two short words from the dictionary and join them with some non-alphabetic character. This leads to some reasonable passwords that are easier to remember than total gibberish, but perhaps they are harder to guess automatically. I am not totally happy with the approach - a regular algorithm must lend itself to regular decryption.

Likewise, I am against password ageing systems - because after a very small number of iterations, users start creating passwords that are 'something easy to remember' rather than 'something that is good at foiling the cracker'. In my case I revert to using some constant stem say `fred` and then cycling a last character, generating `fred1`, `fred2` etc. This is bad, but easy to remember - the ageing system has stopped me using sensible passwords in favour of ones that I can remember.

Recently, people have begun to put their trust in 'shadow password files'. Rather than installing the encrypted password in an easily accessible `/etc/passwd` file, it is stored in a special file. The `passwd` and `login` programs look in this special file rather than the standard password file - the win is that this file is read-only to `root` and so mortals cannot see the encrypted passwords. The programs that require access to the passwords all run as root already, so things are purported to be more secure. The warm feeling of security may cause users to select simple passwords that will be cracked easily should the shadow password file be obtained illicitly. Remember, security by obscurity simply doesn't work.

More reading

If you want to go to the sources for this article you should seek out the following papers:

Robert Morris and Ken Thompson. 'Password Security: A Case History.' *Communications of the ACM,* 22(11): 594-597, November 1979. My version is a copy reprinted in the 'UNIX Systems Manager's Manual' from the 4.3 Berkeley Software Distribution documents printed by Usenix in the US.

David A Curry. *Improving the Security of your UNIX System.* An SRI International report published April 1990. My version came in Postscript form from a UKnet information server.

Daniel V. Klein. 'Foiling the Cracker.' *Proceedings of the UKUUG Summer 1990 conference* pp 147-154; ISBN: 0 9513181 7 9. A good paper bringing aspects of password security up-to-date.

Clifford Stoll. '*The Cuckoo's Egg*'. New York Doubleday 1989. Published in the UK by The Bodley Head, 1990.
ISBN: 0 370 31433 6. A good read .

Change your password to a more sensible one - *NOW.* You know it makes sense.

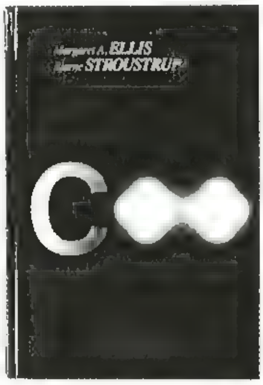
EXE

Peter Collinson is a freelance consultant specialising in UNIX.

Books

This month, the (literally) definitive C++ book, and another approach to the DES algorithm.

The Burgundy Book



The Annotated C++ Reference Manual is a book that many C++ programmers have been waiting for. Here is the complete definition of the C++ language as it stands today, together with annotations and commentaries. Be warned, though: this is not a tutorial. This book doesn't contain any exercises, unlike Stroustrup's original C++ reference manual. If you are not yet a fluent C++ programmer, this book may be an inappropriate place to start.

The book is targeted at language implementors, such as software engineers who wish to develop C++ translators, and at expert programmers who need to check details of the language's definition. If you are unsure whether this is the book for you, the clue is in its title. This is a Reference manual, and it is the most complete definition of the C++ language that you will find, and is up-to-date as of February 1990. It has been adopted as the 'base document' for the ANSI standardisation process for C++.

The manual has 16 main chapters, and two appendix chapters. Each contains an abstract, followed by the reference manual material. The manual sections contain annotations after points that need further explanation, and each chapter is rounded off by a commentary section. The annotations and commentaries serve to tell the reader some of the things that are inappropriate in a reference manual but that are essential for a full understanding of the language and of its designers' intentions. Issues such as the reasoning behind intentional omissions from the language, why features are defined the way they are, and techniques for implementing selected language features all appear in the annotations and commentaries.

The content of the book is straightforward: starting off with an overview of the language, the authors proceed to cover fundamental issues such as lexical conventions, basic C++ programming concepts, and standard data type conversions. They go on to describe expressions, statements, declarations, and declarators. The first detailed mention of classes is on page 163, a little before the book's half way point. Succeeding chapters describe derived classes, member access control, special member functions such as constructors and destructors, and overloaded functions and operators.

Interestingly enough, considering that there is currently some controversy about whether multiple inheritance should be allowed into the C++ standard, this book contains no justifications for, or substantial examples of the use of, multiple inheritance. The reader is pointed towards the relevant papers by Stroustrup and others on the subject.

The next two chapters are exclusively commentary, although they are written in reference manual style complete with annotations. They describe language features that are still experimental and subject to change: templates, and exception handling. Templates are used to define the layout and operations of a set of related classes: new classes that combine the attributes of the template with specific base class types can then be defined by reference to the template.

After the chapter on exception handling, the next chapter is a discussion of preprocessing. The book concludes with a grammar summary, a discussion of compatibility issues that have arisen during the evolution of the C++ language (including a summary of the revisions at each stage and a list of incompatibilities with ANSI C), and a comprehensive index.

Surprisingly (in view of what it is) this book is one of the most readable and accessible programming language reference manuals I have ever read. The annotations and commentaries help make the difference. Some of the annotations are quite amusing, giving insight as they do into the workings of Stroustrup's mind.

Is this book up to date? C++ is a moving target, as the evolution of the language continues. The book makes no direct reference to C++ release numbers. It is up to date at least as far as February 1990 and (although if any eagle eyed readers can correct me they should do so) I believe it to be consistent with release 2.1 of AT&T's CFront.

Should you buy this book? If you wish to learn C++, don't buy this book unless you are already a specialist in comparative programming languages. Buy this book when you think you are nearly fluent in C++, to learn the details and the designers' justifications for specific features. Experts will want this book at their side to verify exactly how the language is defined. This is the only complete definition of today's C++ language: it is also the most readable language reference manual I have ever come across.

Paul G Smith

Title: The Annotated C++ Reference Manual
Authors: Margaret A Ellis, Bjarne Stroustrup
Publisher: Addison-Wesley
Price: £25.95 *ISBN: 0-201-51459-1* *Pages: 447*

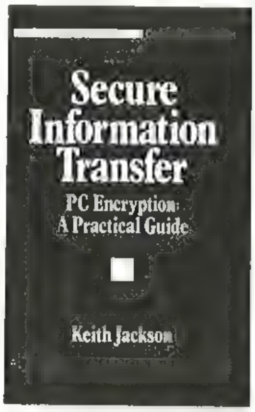
PRIZE COMPETITION!

Our review copy of this book was given to .EXE Magazine by Dr Bjarne Stroustrup himself, on condition that we passed it onto to a needy programmer when the review was complete. We are, therefore, holding a competition, with our copy of the book as the first prize. We will also be handing out .EXE T-shirts to the first three runners-up in this mad, mad, mad, give-away bonanza.

The competition is easy, especially for Peter Greenaway fans. Take a look at our illustration for this month's Third Side article on the POP-11 programming language (Page 71). The brief to Nick Grant, our artist, was that he should include 11 items which suggested the word 'pop' (or derived variants such as 'pops', 'popping' etc). We just want you to list them all.

Answers (together with preferred T-shirt size/design) should be sent to The Editor at the address given on Page 1. Mark envelopes 'POP-11'. The prizes will be awarded to the entries with the most correct items or, in the event of a tie, by drawing the winners' names from a hat. Closing date is 1st of November 1990.

My friend DES



All the computer security books that I have encountered are aimed either at the extreme 'suit' end of the market ('it is probably unwise to leave floppy disks lying around, so be sure to organise a regular tidy up procedure') or are written in maths for the genuine expert. *Secure Information Transfer*, although displaying a few suity signs, is delightfully practical and technical - it concludes with a 60 odd page code listing in Pascal. But I am ahead of myself.

The book's stated aim is to provide a practical guide to data encryption. Chapter 2 provides an overview of encryption techniques, explaining concepts such as symmetric and asymmetric algorithms, use of keys and so on. The RSA algorithm (factorisation of large prime numbers) is briefly described, plus the reason why it is unsuitable for PC work (a single RSA decryption can take many minutes, even running on a powerful model). Mercifully, no attempt is made to explain the theory of the algorithms, or compare effectiveness. The author favours the use of published algorithms over secret ones on the grounds that the former have certainly been scrutinised by cryptographers and survived, whereas the veil of secrecy surrounding the latter may also conceal flaws.

Chapter 3 is all about an old friend: the DES algorithm. In fact, this is only a very brief and general description (although it is backed up by a full implementation given in the Appendix). However, it does include a list of weak and semi-weak keys which should not be used with the algorithm, as they compromise its strength. (Weak keys make the encryption and decryption processes identical; pairs of semi-weak keys have the property that encryption with one key is the same operation as decryption with the other.)

Chapter 4 is devoted to describing a selection of commercial PC data encryption products, software and hardware. The action of each package is given, plus the type of algorithm used for encryption. The author also notes strengths and weaknesses of the different systems, for example, a file encrypting program which always overwrites the original text is highlighted.

Chapters 5 through to 8 all concern the DES-LOCK program. This is a DES file encryption program, written and marketed by the author. It is a slightly cut-down version of DES-LOCK that is supplied in (Turbo Pascal V3.0) code form at the back of the book. It is quite full-featured, so that Chapter 5 contains a brief indication of its capabilities, Chapter 6 outlines the structure of the program, Chapter 7 is a huge, 42-page user manual (presumably the manual of his commercially released version filched virtually intact; it starts off by recommending that you back up your master disk containing the .COM file - surely unnecessary advice if you have just typed in 60 pages of Pascal) and Chapter 8 provides a few examples of use.

Appendix A contains the DES-LOCK source code, as already mentioned. Appendix B lists the reference ANSI and ISO standards for DES. There is also an ASCII chart and a bibliography.

Whether you need to incorporate encryption security into one of your programs, or just require a guide to what there is and what it does, you should find this book just the ticket. I only hope the author makes enough on it to buy himself a new Pascal compiler.

Title: Secure Information Transfer *Author: Keith Jackson*
Publisher: Blackwell Scientific Publications
Price: £29.95 *ISBN: 0-632-02664-2* *Pages: 176*

THE ATARI ABC

Now, there's a PC-AT compatible that not only solves problems like other AT compatibles, it also solves the one problem that its predecessors have created ... affordability.

Silica Systems are pleased to present the new Atari ABC. The ABC is a 286 AT compatible that runs at over 5 times the performance of the IBM XT. This is achieved by combining the power and speed of a 16-bit 286 processor, with Atari Corp's development experience and engineering capabilities, which use the very latest in design technology. However, the Atari ABC's are available at 'XT prices' and many XT owners will wonder why they paid more, but got less!

The ABC is a reliable, high performance computer, built to exceptionally rigorous standards. It is well designed, to a state of the art specification, maintaining maximum expansion capability for the future. This includes up to 4Mb of RAM and 3 AT expansion slots. Plus, unusually for a PC at this price, the ABC has the ability to install two or three extra drives, with the cabling already inside the CPU.

The ABC has a host of impressive features, all built-in as standard, encouraging simple installation and ease of use. In addition, the ABC's small footprint and quiet operating, mean that you will notice it less on your desk than other PC's. Except, of course, when you begin to take advantage of its AT power.

At its remarkably low price point, the ABC is ideally suited for home, office and educational use and will take full advantage of the vast range of PC compatible software. And, with 12 months FREE on-site maintenance (8 hour response), you can rest assured that your ABC will be fully supported.

The ABC is the latest addition to a comprehensive range of both 286 (8Mhz or 12Mhz) and 386 (16Mhz and 20Mhz) PC's, from the Business Systems Division of Atari Corp. Atari's PC range offers an unsurpassed combination of reliability, compatibility and expandability, with unique features like Atari's 44Mb removable hard disk on their PC4. Return the coupon NOW for further details of the complete range of PC products from Atari's Business Systems Division.

ABC TURBO
AVAILABLE: JUNE 90

To put the ABC even further ahead of the competition, we have introduced a special 'Turbo' version, which includes a super fast RLL hard disk controller to increase the data transfer rate from 150KBytes/s to 800KBytes/s, with a performance index of 4.801. Plus! to provide exceptional graphics with an outstanding colour resolution of up to 1024×768, we have included a Super VGA graphics adaptor and a 14" VGA colour monitor (which also supports 1024×768 resolution), all for an additional price of only £300. Check the ABC Turbo against the competition, at only £1299 (+VAT) with 30Mb hard disk, Super VGA colour graphics capabilities, plus 12 months on-site maintenance, it offers unbeatable value.

LOW PRICE - HIGH SPEC!

CPU+MOTHERBOARD
- 8Mhz 80286 16-bit CPU
- Landmark Speed = 10.3Mhz, Norton SI Test = 9.0
- Latest NEAT (New Enhanced AT) technology chip set
- Has user selectable bus, DMA speeds & wait states
- Programmable to operate at 0 or 1 wait states
- 640K RAM base memory (expandable)
- Motherboard expandable to 4Mb RAM memory
- Real time clock with battery back-up
- Full 16-bit data line capability
- Socket for 80287-8 numeric Co-Processor
- 3 16-bit AT expansion slots (max)

DISK
- 32Mb RLL hard disk - 150KBytes/s transfer - 60ms access
- Fast disk controller - 800KBytes/s transfer - *Turbo version*
- 1.44Mb built-in 3½" floppy disk drive
- Space for two 5¼" and two 3½" drives (max)

I/O
- Parallel port built-in (25-pin Centronics)
- Serial port built-in (9-pin RS-232)
- Mouse controller port built-in (9-pin serial)

GRAPHICS
- Super VGA video graphics controller - *512K - Turbo vers*
- Super VGA: 256 colours in 800×600 res - *Turbo vers*
- EGA video graphics controller built-in
- EGA: 16 colours available from a palette of 64
- EGA: 256K of dedicated Video RAM built-in
- MDA, CGA and Hercules graphics controller built-in

PHYSICAL
- Low noise operation
- Ergonomic small footprint (38cm×41cm×15cm)
- Keyboard - 102 key enhanced AT (12 function keys)
- Keyboard - has 2 position height adjusters

PLUS!
- Fully OS/2 compatible (requires 4Mb RAM)
- Supports Industry Standard Networks
- MS-DOS 3.3 with GW BASIC 3.22
- 12 Months FREE on-site maintenance included

286 AT

WITH 30Mb HARD DISK & EGA MONITOR - £899

FREE! - ON-SITE MAINTENANCE
FREE! - TECHNICAL SUPPORT HOTLINE
FREE! - TEST/BURN-IN BEFORE DESPATCH
FREE! - OVERNIGHT COURIER DELIVERY

£899 +VAT WITH **30Mb HARD DISK** AND **EGA MONITOR**

ABC TURBO £1299 EXC VAT
- *SVGA COLOUR MONITOR*
- *SVGA GRAPHICS CARD - 512K*
- *30Mb HARD DISK DRIVE*
- *FAST DISK CONTROLLER*

£899 +VAT= £1033.85

ATARI

	3½" FLOPPY DRIVE NO HARD DRIVE	3½" FLOPPY DRIVE + 30Mb HARD DRIVE
NO MONITOR	£599 +VAT = £688.85	£799 +VAT = £918.85
+ EGA MONO	£699 +VAT = £803.85	£899 +VAT = £1033.85
+ EGA COLOUR	£799 +VAT = £918.85	£999 +VAT = £1148.85
ABC TURBO AVAIL JUNE 90 • 14" SVGA COLOUR MONITOR • SVGA GRAPHICS ADAPTOR • FAST DISK CONTROLLER	£1099 +VAT = £1263.85	£1299 +VAT = £1493.85

SILICA SYSTEMS OFFER YOU

ON-SITE MAINTENANCE (8 hour response): Available Free of Charge on all ABC's.
TECHNICAL SUPPORT HELPLINE: Team of Atari technical experts at your service
100% QUALITY CONTROL: Every ABC is tested/burnt-in before despatch.
FREE OVERNIGHT COURIER DELIVERY: On all hardware orders shipped in the UK.
ESTABLISHED 12 YEARS: Proven track record in professional computer sales.
£13M TURNOVER (with 60 staff): Solid and reliable with maintained growth
BUSINESS/EDUCATION/GOVERNMENT: Volume discounts available for large orders.
PRICE MATCH: We normally match competitors on a "Same product - Same price" basis.
SHOWROOMS: Demonstration and training facilities at our London & Sidcup branches.
THE FULL STOCK RANGE: All of your Atari PC requirements from one supplier.
FREE CATALOGUES: Will be mailed to you with offers and software/peripheral details.
PAYMENT: Cash / cheque / credit card / credit - we are licensed credit brokers - details on request.

Before you decide when to buy your new Atari ABC, we suggest you think very carefully about WHERE you buy it. Consider what it will be like a few months after buying your ABC, when you may require additional peripherals or software, or help and advice with your new purchase. And, will the company you buy from contact you with details of new ABC products? At Silica Systems, we ensure that you will have nothing to worry about. Silica Systems is a new division of Silica Shop, the UK's leading Atari specialists. This new division has been established to provide a service to the more serious home user, as well as to business and education purchasers. Silica have been established for over 12 years, and have an annual turnover of £13 million. With our unrivalled experience and expertise, we can now claim to meet our customers requirements with an understanding which is second to none. But don't just take our word for it. Complete and return the coupon now, for our latest Free literature and begin to experience the "Silica Systems Service"

SILICA SYSTEMS

MAIL ORDER: 1-4 The Mews, Hatherley Rd, Sidcup, Kent, DA14 4DX Tel: 081-309 1111
Order Lines Open: Mon-Sat 9.00am-6.00pm No Late Night Opening Fax No: 081-308 0608

LONDON SHOP: 52 Tottenham Court Road, London, W1P 0BA Tel: 071-580 4000
Opening Hours: Mon-Sat 9.30am-6.00pm Late Night: Thursday until 8pm Fax No: 071-323 4737

SIDCUP SHOP: 1-4 The Mews, Hatherley Rd, Sidcup, Kent, DA14 4DX Tel: 081-302 8811
Opening Hours: Mon-Sat 9.00am-5.30pm Late Night: Friday until 7pm Fax No: 081 309 0017

BUSINESS/EDUCATION: 1-4 The Mews, Hatherley Rd, Sidcup, Kent, DA14 4DX Tel: 081-308 0888
Order Lines Open: Mon-Fri 9.00am-6.00pm Closed on Saturdays Fax No: 081 308 0608

To: Silica Systems, Dept EXE-0690-28, 1-4 The Mews, Hatherley Rd, Sidcup, Kent DA14 4DX

PLEASE SEND INFORMATION ON ATARI PC's

Mr/Mrs/Ms: .... Initials: .... Surname: ....

Address: ....

.... EXE 7/90

.... Postcode: ....

Tel (Home): .... Tel (Work): ....

Company Name: ....

Which computer(s), if any, do you own? ....

E&OE - Advertised prices and specifications may change - please return the coupon for the latest information.

CIRCLE NO. 300

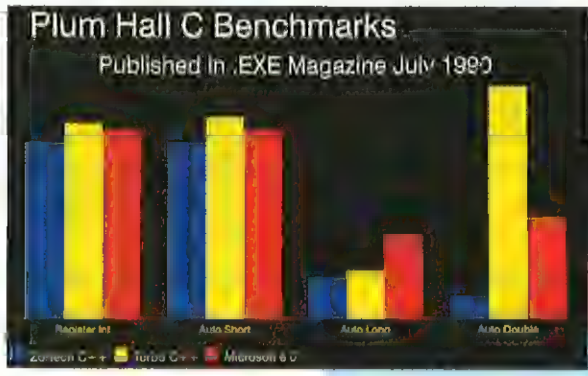
Multi - Platform

MS-DOS • WINDOWS • OS/2 • DOS 386 • UNIX 386

MS-DOS

Zortech's industrial strength compiler provides all the benefits of C++, but with the speed and code size you would expect from the best C compilers.

The quality of the original Zortech C++ implementation together with the continuous improvement achieved since its launch in June 1988 produces fabulous benchmarks. Just look how far it's ahead of the nearest competitor.



Zortech C++ provides state of the art, USEFUL features, most of which are added in direct response to customer requests.

You can effortlessly cruise through the DOS 640K barrier using Zortech's Virtual Code Manager (VCM™). This allows you to develop applications up to 4MB in size whilst in real mode, without changing your C/C++ source code. Zortech's much acclaimed 'handle pointers' provide an elegant solution to processing EMS memory.

Zortech C++ also uses the Rational Systems™ DOS Extenders allowing you to easily compile and debug really large programs, even large MS-Windows 3.0 applications. If you want to purchase a Rational Systems license for your own applications, your Zortech code is Plug & Go.

Zortech's new C++ Workbench provides a cross platform development environment for C++. It has really useful features including powerful source and grep browsers, to look at your handiwork.

In response to hundreds of requests, MS-Windows 2.1 support was added into the base DOS C++ Compiler in version 2.0. Now with Zortech C++ V2.1 development of C++ applications for Windows 3.0 is a reality not a promise.

Along with the C++ compiler comes a top quality ANSI C compiler. In fact, after reviewing 14 C/C++ compilers in its May 1990 issue, Computer Language editor J. D. Hilderbrant said:

"The pressure to name an overall winner in the compiler sweepstakes is nearly overwhelming...it's an easy choice. We pick Zortech!"

Thousands of our customers had existing C code they wanted to recompile, so we made it simple. In the words of BYTE Magazine:

"I fed a Microsoft C specific version of the Micro-EMACS editor source to Zortech's compiler, and less than one hour later, I had a new (and smaller) program."

Our C++ Debugger, which understands C and Assembler too, is CodeView™ compatible, but that's where the similarities end. This feature packed tool can examine your program from 19 viewpoints and uses overlapping windows with full mouse support, icons and dialog boxes.

Debugging large programs is no problem with our DOS Extender, Virtual and Remote debugger versions. Quite simply, there's no better C++ debugger to use and no better C++ to debug.

Our C++ Tools package is the most comprehensive set available. All 25 class libraries are extensively documented and come with the full source code.

The Zortech C++ Developer's Edition V2.1 includes C and C++ Compilers, C++ Debugger, C++ Tools and the FULL Library Source Code (excluding Flash Graphics). That's right, you don't have to pay hundreds of dollars extra for source code - it's in the box!

MS-WINDOWS

Improved support for MS-Windows (including new Windows 3.0 support) is provided in the base C++ DOS compiler, at no extra cost. With Zortech, you can now even compile from within Windows!

Support for new extended keywords _loadds and _export as well as the ability to create DLL's make programming in Windows with C++ practical. We provide extensive documentation and 50K of sample code to illustrate development of

ZORTECH Inc., 4-C Gill Street, WOBURN MA 01801 Tel: 617-937-0696 Fax: 617-937-0793 Orders: 1-800-848-8408
ZORTECH Ltd., 58-60 Beresford Street, LONDON SE18 6BG Tel: +44-81-316-7777 Fax: +44-81-316-4138

ZORTECH C++ DEVELOPER'S EDITION V2.1

Zortech C++

MS-DOS • WINDOWS • OS/2 • DOS 386 • UNIX 386

applications in this exciting new environment. Do you need MS-Windows class libraries? Call for details of third party Zortech Validated Products.

OS/2 NEW

The OS/2 Developer's Edition now provides a C++ Compiler and source level Debugger designed for C++. In the words of OS/2 Magazine:

"Zortech C++ serves as a direct replacement for the Microsoft C Compiler in developing applications, allowing programmers to use object-oriented techniques in OS/2 development."

The OS/2 Developer's Edition also includes C++ Tools, Flash Graphics and C++ Workbench for OS/2 together with the standard DOS Developer's Edition.

Upgrades for existing OS/2 Compiler Option owners now available. Please call for details.

UNIX 386 NEW

Not a day passes at Zortech HQ without numerous requests for a UNIX version of Zortech C++. Now, DOS and OS/2 developer's can reach new markets by easily moving their code to SCO UNIX 386 and binary compatibles.

The Zortech C++ V2.1 UNIX 386 Compiler generates the same tight, fast code that Zortech's DOS and OS/2 users have come to expect. UNIX specific versions of Flash Graphics and the C++ Workbench are also provided.

In line with the traditional Zortech Policy, owners of the Zortech C++ V2.1 UNIX 386 Compiler will be able to inexpensively upgrade to the forthcoming Zortech C++ V2.1 UNIX 386 Developer's Edition.

DOS 386 NEW

Now, with the 386 you can address up to 4 Gigabytes of memory. Why spend so much money on 386 hardware and not use software which will take advantage of it?

On the other hand, you need to retain the facilities of standard MS-DOS too.

MS-DOS developers can now build true 32 bit C and C++ applications for 386 processors using Zortech's powerful 386 development system. The Zortech C++ V2.1 Developer's Edition for DOS 386, contains 32 bit versions of the C and C++ Compiler, Flash Graphics library, C++ Debugger and full standard library source code together with all the familiar features provided with the standard DOS Developer's Edition.

Using Phar Lapp's much acclaimed 386/DOS Extender Technology, you can build applications which access 4 Gigabytes of linearly addressable memory. Your applications will also be Plug & Go for use with Phar Lapp's 386 DOS Extender which may be purchased seperately.

C++ VIDEO COURSE

Zortech's C++ Video Course is all the training material you need to move a team of good C programmers into the world of C++. Many corporations have already done just this.

Cut the hotel bills, travel expenses and fees of outside training courses and seminars - not to mention the inconvenience and disruption to your normal routine.

Use a proven training tool, that in one hour a day, over a period of six weeks, can train your whole team in C++ for the price of one airline ticket.

The course consists of 32 tutorials on six one hour VHS tapes together with one 256 page workbook containing course notes and exercises. Unlimited additional course workbooks are available at modest cost. Compiler & hardware independent. NTSC or PAL format available.

C++ at Work '90

The new Zortech C++ platforms will be officially released in New Jersey during C++ at Work '90 Sept 24-26, 1990.

CIRCLE NO. 301

THE RECESSION, AND HOW TO SURVIVE IT

Want to stay in a job throughout the coming recession? Go and work for a supermarket chain installing off-the-shelf packages on PC networks.

So says software business guru Robin Bloor of publishing house ButlerBloor. The reasons are simple according to Bloor. Food suppliers don't suffer from bad times as much as the rest of the business world, because people still have to eat, and packaged software already accounts for 70% of the of the market.

While the future looks bleak for development staff with skills in moribund languages such as COBOL, Boor says that he foresees a healthy future for C-dogs. 'This is a skill that is going to grow. If people are going to be badly affected by recession in the employment area then it's going to be the dying languages and environments that are going to be worst hit.

'I would have thought that COBOL skills now are of no particular merit.'

ButlerBloor produces a series of reports on the best selling packages for systems ranging from PCs to mainframes, so Robin Bloor gets to see inside companies and find out their plans for the future.

Hence his prediction that packaged software is set to sweep the board. The PC and workstation market has 'grown like crazy' he says.

'But there's not a lot of that market which needs development work. There are some really quite sophisticated packages now. You might, for instance, need to program graphical stuff, but the odds are that the system will be a package.'

Robin Bloor foresees a bad two years ahead for the computer industry as projects get cut. But he believes that the industry, by its very nature, will not suffer as badly as the obvious contenders for economic disaster like the building trade. Computers are partly recession-proof as companies often install systems in a desperate attempt to save money.

Because of the trend towards off-the-shelf solutions, Bloor believes that bespoke software houses will be badly hit by the recession. 'The business will still require people,' he says, 'But it won't require software houses.'

So where should you be working if you want to stay safe? Well, it's back to the supermarket chains.

'They aren't badly affected by recession. They'll drop by maybe 5% because no matter how bad things get, people still have to eat. People also drink and smoke, so alcohol and tobacco companies are safe.'

Projects in those areas will probably not get cut, along with those in areas like sophisticated engineering and areospace.

'But small companies will go to the wall. Not because they're any worse than large companies, but because that's just the way things happen.'

But the safest job of all is the dullest in the software career guide: maintenance.

Maintenance accounts for 70% of the IT budget in some highly computerised corporations, says Bloor, so there is only 30% the company can safely trim.

PSYCHO!

When a prospective employer next tells you he wants to pick your brains, beware. He may not just want to know what you think of the company. He will probably want to test your personality.

Psychometric testing, to give the practice of personality vetting its proper name, is on the increase in the UK. Its fans claim that it helps the selection process and prevents the personnel manager's nightmare: - that, three years on, the recruit who looked as though he was 'just having trouble settling in' is still defiantly refusing to fit his square peg into the company's round hole of a job.

Opponents of psychometric testing say that it boils down to whether the job-seeker was in a good mood or a bad mood, a little hungover or, indeed, whether they understood the question. Many psychometric tests are about as clear as an Inland Revenue form.

One American business psychology firm markets a test that - according to the blurb - will uncover tendencies towards dishonesty. It's so cunningly constructed that you can't even fib your way through it.

Questions in personality tests ask you to agree or disagree with rather bland statements, such as 'I prefer walking to reading', and sometimes rather surrealistic ones like 'You find a turtle on its back in the desert, do you turn it over?' The test scorer can then measure your responses against indexes for introversion/extroversion, confidence/lack of self esteem and so on.

Business psychology companies spend wads of money on marketing their tests, and not all are as kosher as they appear. Recruitment expert Philip Schofield has collated a list of "dos and don'ts" to help the wary.

If the test contains American spellings or phrases, do ask whether it has been validated for use in the UK. Somewhat unsurprisingly, Americans tend to average much higher scores for extroversion than do Brits. If the test is home-grown, ask if it has been developed by a member of the Psychological Society's Division of Occupational Psychology. If it hasn't, the chances are it's a turkey.

If the test asks about your sexuality, phobias or feelings towards your parents, it's probably intended for clinical use - and anyway it's unethical. Protest. Finally, if it's an ink blot (Rorschach) or picture test, then it's a clinical test again. This is a waste of both your time and the boss's. Incidentally, if you go for a job with a French company they will almost certainly pooh-pooh the idea of psychometric testing and give you a handwriting test instead. Start practising that joined up handwriting!

OUT OF RETIREMENT

Whiz-kid coders look out. A new firm based in Portsmouth is hoping to sell the expertise of ex-IBMers.

Skillbase has been launched to use the knowledge of IBM employees who have to quit to take early retirement. Employees will work as consultants in their field of expertise, whether it be logistics, personnel or, gulp, programming.

Skillbase workers are guaranteed 90 days pay, although they will be paid more if they work for longer. The firm hopes to take on non-IBMers as well.

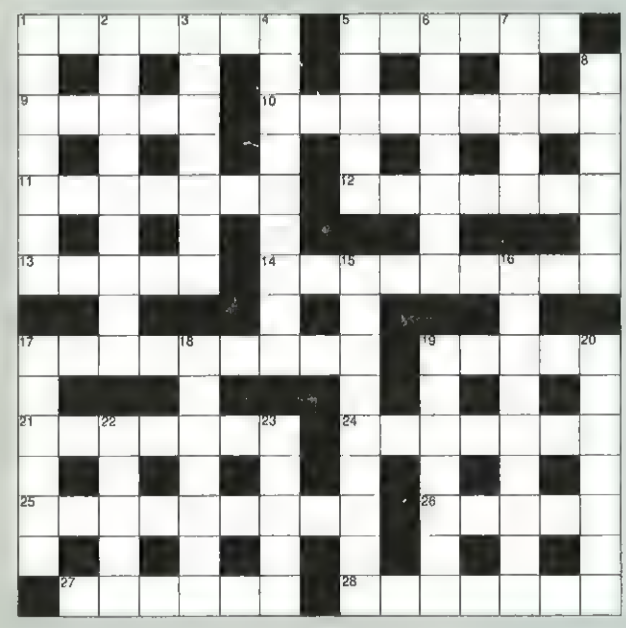
The move comes after IBM has been trying to slim down by encouraging early retirement among its work-force. The company also believes that Skillbase will act as a useful stepping stone for early retirers - who can be as young as late 40s - who want to move into a different career.

UNPOPULAR SWISS ROLL

Swiss firm Schindler Lifts is opening an R&D centre in Livingston, Scotland, because it can't persuade software engineers to work in Switzerland. The firm hopes to install 20 techies in its new site by Christmas. Schindler usually recruits programmers from abroad and relocates them to Switzerland, but has found this difficult going - so has moved into Scotland to tap the local pool of talent. *Jim McGuin*

EXE

.EXEWORD OCTOBER

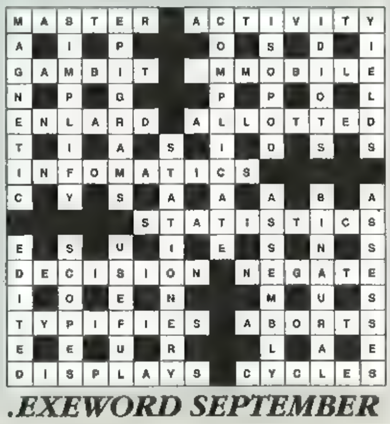


ACROSS

1&5 Distributes paper outputs from a program such as... (7,6)
9 ...the one whose fruit makes one dream (5)
10 Simulation for Page 3? (9)
11 Grass hopper I hear (7)
12 Determine to get the answer again (7)
13 Kept-bacon by recording the program (5)
14 Male or female? Very much so! (9)
17 Drums of data (9)
19 Micro coming up in the spring (5)
21 Journal for old calculators (7)
24 Remove it from a disc to make a contract, they say (7)
25 Transferred a character and passed you in France (5,4)
26 Screen mnemonics for wimps (5)
27 Character on the world stage of computer games? (6)
28 Intellectual in the nest perhaps (7)

DOWN

1 Base of wafers, quartz, agate, sand... (7)
2 Getting hold of lost data? (9)
3 Poor person's PC, some say (7)
4 Sort of binary code of ancient telegraph (9)
5 Someone off the fencewith a drink it seems (5)
6 Takes on staff two ways and gets ordered output (7)
7 Early run for the judge (5)
8 Shook hands with an appetite (6)
15 Not here - in hospital? (9)
16 Instrument that gets hammered (9)
17 Writer's corner - lots in 9 (6)
18 Some such sun sees mad and Englishmen (7)
19 Working through a program to get an image on paper? (7)
20 Read with care through old second-hand... (7)
22 ...hammer for the chair (5)
23 Data entry clerk with crucial field shows sign of hesitation (5)



.EXEWORD SEPTEMBER

'C', C++, UNIX AND MS-DOS

LONDON, HOME COUNTIES, THAMES VALLEY

SALARIES: £15-36,000

As one of the leading names in software staff recruitment, Logistix has a vast number or vacancies which cover most of the technically interesting areas in the industry. Our current requirements are for Software Engineers at various levels ranging from Programmer to Team Leader level with a background in software development in any of the following:

- **MS/Windows or X/Windows**
- **HYPERTEXT APPLICATIONS**
- **NON-STANDARD GUI'S**
- **RELATIONAL DATABASES**
- **OBJECT ORIENTATED PROGRAMMING**
- **GRAPHICS APPLICATIONS**
- **NETWORKING**
- **FINANCIAL SYSTEMS**
- **DEALING ROOMS**
- **IMAGE PROCESSING**

If your experience, or interest, lies in any of the above or allied areas then call us on the number shown below quoting REF.EX/AS.

We will be able to advise you on all aspects of your career development and put you in touch with some of the most interesting projects currently being undertaken in this country.

Logistix Recruitment Limited
Logistix House 148 King Street
London W6 0QU
Fax 081 563 0166

081 741 8511

Logistix

ADVERTISERS INDEX

STOB - Return to Tantrum Towers

BITE Magazine celebrates its 15th anniversary this month. Stob invited Professor Hunter Rubbish, the famous BITE columnist, to look back over the years of his contributions.

Hi! Wow. Yeah. Wow. Uncle Hunter calling. It's been one heck of a month. The last two weeks I've spent in continuous faxtact with NASA, okaying the blueprints for the Shuttle/2; I can reveal that their plans are pretty awesome, and follow amazingly closely the lines predicted in my novel *Crisis at the Doom of the Cryptic Void* and its sequel *Ted the Space Ranger Rides Again* (Rubbish Books, $12.95 apiece and a bargain), so that was nice.

Anyway, I must get on. Little Verity Blob has asked me to do a quick resumé of my time here at BITE Magazine and, since I had a three minute window, I said that I would be delighted.

Of course, I joined BITE Magazine in the very early days, when its founders, Steve McGraw and Steve Hill, were still turning it out from a small rabbit hutch in Alabama. I guess we were all pretty young and inexperienced - I remember taking turns to burp Steve M after meals - but what we lacked in knowledge we made up.

I well remember the first sentence of my first Tantrum Towers column and, what do you know, I think it is as relevant now as it was then. Here it is: 'Hi!' As I often say to Mrs Nigella Rubbish, my wife and horse, 'Always connect' (the wise words of EM Forster, the cable manufacturer).

There were some *real* characters at the first annual BITE Editorial Conference. You young bucks may not remember pioneers such as Big Zeke, Dwayne the Monkey and, of course, the sinister Cray twins: Steve and Olly. Sometimes I wonder whatever happened to all my old computers. Nigella used to use Big Zeke for her comms work - she's chairmare of the hay conference on BIX - right up to last year, when we finally shipped him off to his final resting-place in the Museum of Very Important People's Personal Belongings, California.

Not all of my predictions were exactly right, but most were. When I wrote, 'MS-DOS sucks; it will never sell - that little punk Gates will be filing for bankruptcy before the month is out', a lot of people got hold of the wrong end of the stick. It was a purely *technical* criticism, which I stand by to this very day. On the other hand, I do flush to recall my suggestion that the distribution in the USSR of the Russian edition of my book *Time Dreams of the Golden Warriors* would provoke a counter-revolution by the end of 1985. As we all now know, with 20-20 hindsight, the Soviet ice did not begin to melt until mid-1986.

It's the darndest thing: I've filled up Miss Blob's little hole already, and not said one tenth of the things that I meant. Finishing up quickly: this month's piece of hardware that I've been given and you can't possibly afford is an electron microscope. It's got nothing to do with computers, but its worth more than a hundred thou, so what the hell. Here's to the next 15.

Note to Editor: What does '.EXE' mean anyway? N says it's something to do with computers, but I figure it's the name of the fifth Teenage Mutant Ninja Turtle. HR.

EXE

.EXE
Please mark 'Later' if appropriate
Notify 050 SAH for address change

		Date Seen
~~EC DISPLAY 2 WKS~~	~~LIBRARY, EC~~	30.10.90
~~PJ WINFIELD~~	~~HERB/VEG, EC~~	
~~DG ROSE~~	~~PLANT HEALTH, EC~~	
~~D MARR~~	NEM/~~ENT, EC~~	
EC LIBRARY	EC	

Please remove my name D.G. Rose

02686872/V5*N5/EC.S/1

Taxes
Software Piracy

We can save you from one of them.

Sorry. Death we can't do anything about. As for taxes, when you use our product you'll probably wind up paying more. But software piracy: there we offer some help. Our family of software protection devices (dongles) have improved unit sales for over 2,000 companies around the world. Our products can be used in the MS-DOS, OS/2 and Macintosh environments.

Build Your Own Custom Protection Environment

Use our patented "dual-locking" ASIC chip as the basic building platform. Next, add options like: on-the-fly read/write memory, write-once or multiple-write locking codes, and encryption shells. Then add your own programming creativity to build a protection environment best suited to your product.

Users attach the device to their parallel port, and programs won't run without it. Back-up copies, hard disk and LAN operation are not interfered with.

Your Intellectual Property Belongs To You

And if you don't protect it, who will? Our products offer the most equitable way to protect your interests without sacrificing the rights of your customers. Call us today for information and demonstration units.

Software Security UK Ltd.

21A The Precinct
High Street
Egham, Surrey, UK TW20 9-HE
0784 430 060 FAX 0784 430 050

CIRCLE NO. 303